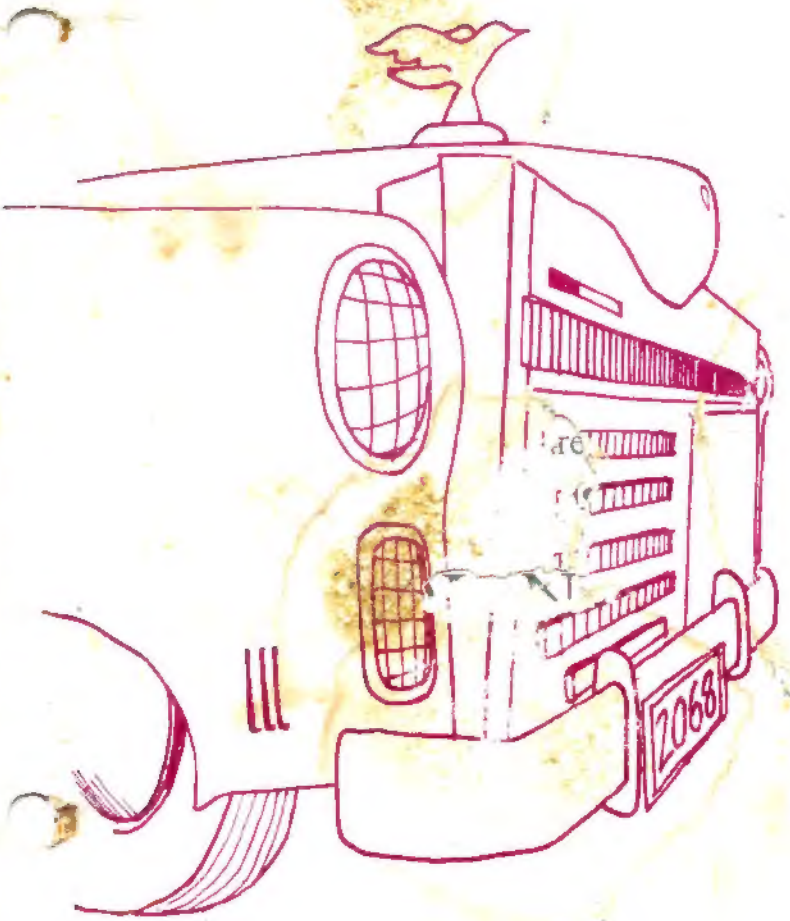
MARCH/APRIL '87
Vol. 3 No. 3
$3.00 U.S. Funds

AMERICA'S LARGEST TIMEX SINCLAIR MAGAZINE

TIME DESIGN MAGAZINE



Classy Front End

A TDM EXCLUSIVE
BY PAUL BINGHAM!

Sir Clive's Pandora Arrives!

*** NEW LIFE FOR THE ZX81/TS1000 ***

SILICON MOUNTAIN COMPUTERS announces TRUE HIGH RESOLUTION SOFTWARE for the ZX81/TS1000. You read it right! Without any expensive hardware add-ons, your computer can now run software that even its designers never dreamed possible. Thanks to an amazing discovery by Wilf Rigter, and innovative programming by Gregory Harder and Fred Nachbaur, you no longer have to suffer the "low res blues." Multiple character sets, 256x192 graphics, 64-column screens, UDG's, even SPRITES are now available for your computer!

NO computer modifications are required. If you have a ZX81, TS1000, or TS1500, with a 16K (or larger) RAM pack, plus an 8K static RAM board, you already have all it takes to run this remarkable software. Suitable static RAMs include the popular "Hunter" board, or similar designs (see SyncWare News vol. 4 no. 1 for one such project costing under $10). Alternately, consider our 8K "SCRAM" board (described below).

All prices include shipping in USA and Canada. Foreign orders: please add $5 for air shipping. CDN$ accepted at par from Canadian customers. Write for catalog of other available software. At SILICON MOUNTAIN COMPUTERS, the ZX81 family of computers is our ONLY specialty. Our goal is to develop the most progressive software ever created for these machines. We feel that the software listed below propels these machines into mainstream computing; we think that you'll agree.

"SCRAM" NVM BOARD

With the loss of the "Hunter" board from the market, we saw the need for an improved functional equivalent, at a lower price! This board works with ZX81/TS1000 or TS1500, has on-board battery back-up protection, and supports all of our high-res software with no modifications. It can be mapped in 0-8K (ROM overlay), 8-16K (normal operation), even 16-24K or 24-32K.

Other features include:

- DIP switch to deselect 2K blocks
- Board enable switch
- Write-protect switch
- RESET switch easily installed (optional)
- Very low power drain
- Feed-through connector
- FULLY ASSEMBLED! Just plug it in.
- Use with other machine-code software

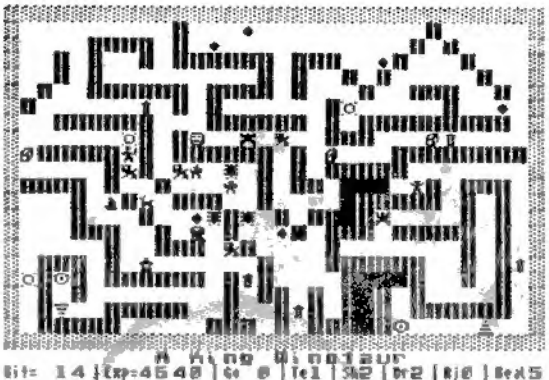
PRICE: $39.95 including shipping.

DUNGEON OF YMIR

This D&D-style graphic adventure game is the most challenging, complex and spectacular entertainment software ever written for the ZX81 family of computers. 9 levels, 124 monsters (16 types), 5 spells, 15 objects. VERY addictive, VERY challenging, VERY impressive graphics. 100% machine-code, yet takes up all available memory in 16K. One reviewer calls it "5-star software" and "a must have." We know that you'll agree.

***** NEW LOWER PRICE *****

PRICE: $19.95 including shipping



YEAR-AT-A-GLANCE

This calendar/appointment book program demonstrates the power you have using SRAM HI*RES Extended BASIC. Enter, update, delete, list, print messages and reminders for any day of any year 1800-2099. Program structure will remind you of much larger machines which shall remain nameless....

PRICE: $9.95 including shipping.

*** SEE SPECIAL OFFER BELOW ***

SRAM HI*RES EXTENDED BASIC is the flagship of our new line of software. With this remarkable package YOU can write high-resolution applications... ENTIRELY IN BASIC! While using only 4K of memory, SRAM HI*RES adds 38 new hi-resolution commands. If you know how to program in Sinclair BASIC, you will find SRAM HI*RES easy to learn and use. A revolutionary syntax system allows ANY variables or expressions to be used in your commands. No REMs to pass parameters! No POKEs! A single USR call is used for ALL commands! Most commands can be chained into MULTIPLE STATEMENT LINES! We even included a fast (8.5x normal) set of tape routines! The most reliable tape system ever written. Other features:

- Three 32-column PRINT modes
- Lower-case and new symbols
- 64-column PRINT mode
- 128 User-defined characters
- Scroll WINDOWS any direction... a pixel at a time!
- Up to 32 TRUE sprites! Speed adjustable.
- Invert windows or entire screen
- Software-only video reverse
- PLOT, DRAW, CIRCLE, RECTANGLE, TRIANGLE
- TS2040 printer supported
- Much more!

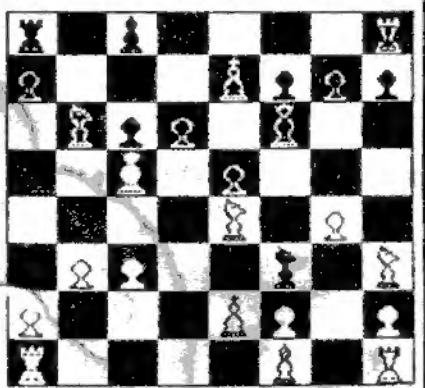
PRICE: $24.95 incl. complete manual & shipping

HIGH-RES CHESS

This program upgrades the popular Psion "CHESS" program (as sold by Timex) to spectacular hi-res format. Included with the tape is an easy-to-follow info sheet detailing how to modify (and back-up) the original program. No longer any need to play along using a "real" chess board!

PRICE: $9.95 including shipping.

	YOU	ZX-81
01	B2-B3	E7-E5
02	C1-B3	B8-C6
03	A3-F8	E8-F8
04	G1-H3	C8-F6
05	D2-D4	C6-D4
06	E2-E4	F6-E4
07	...B3	E4-C3
08	C-B3	C3-B5
09	...C3	B8-F4
10	B3-C4	E7-C6
11	G2-G4	B4-F3
		CHECK
12	E1-E2	B7-B5
13	A3-B5	B5-B6
14	C4-C5	F8-E8
15	B5-D4	E8-E7
16		
17		



SPECIAL OFFER

If you purchase our "SCRAM" NVM board and SRAM HI*RES EXTENDED BASIC and mention this ad, you'll get a FREE copy of YEAR-AT-A-GLANCE. Offer good only until July 1, 1987.

The Peak of Quality... from SILICON MOUNTAIN COMPUTERS

C-12, Mtn. Stn. Group Box, Nelson BC V1L 5P1, CANADA (604)352-1668

The Editor's Forum

First, without sounding like a lot of hype, I would like to encourage as many of you that can make it, to attend the Timex Sinclair Computer Fest in Indianapolis this coming May. Frank Davis and crew have spent an enormous amount of volunteer time and even personal investment, to bring you the largest assemblage of TS vendors and enthusiasts ever seen. I'm hoping that we can show our appreciation by surpassing their anticipated attendance goal. For further details, please check out the news section in this issue.

I have received a number of requests for info on where to get repair work done on Sinclair computers. This made me realize that TDM hasn't reported on this since one of our early back issues.

Currently, we know of three sources for repairs of the TS1000/TS1500/TS2068: Carver Technologies(Tim Carver is the service technician), 3832 Watterson Ave, Cincinnati, OH 45227, tel.(513) 271-5575; Timex Product Service Center, 7004 Murray St., Little Rock, AR 72203, tel.(501) 372-1111 [yep, they still provide "out-of-warranty" service at this time]; Sunset Electronics, 2254 Taraval St., San Francisco, CA 94116, tel.(415) 665-6161.

For QL service, contact either A+ Computer Response (69-B Island St., Keene, NH 03431, tel. 603-357-1800); or Brice Road Pharmacy (1653 Brice Road, Reynoldsburg, OH 43068, tel. 614-861-3600). However, I recommend that you contact the dealer where you purchased the computer from first.

For Spectrum repairs, we have no recommendation, other than consult the back pages of ZX COMPUTING, where several European repair houses are listed.

Another source of help for that faulty computer might be a new regular feature here in TIME DESIGNS (see elsewhere). It's called the "TS Communique", and is hosted by our own Joe Williamson. Joe, as you may know, was the former editor of SUM. He also studied electronics and earned a degree from Florida State, and is currently employed as service technician for a video store. The TS Communique allows you the reader to send in questions about troubles you are having with your equipment (including monitors, printers, interfaces, storage devices, etc.). Answers will be printed in upcoming issues.

In closing, I would like to do something that I've never done before in TDM. Let me explain. A couple of months ago, I was having one of those perfectly rotten days (you know, the ones where nothing goes right?). Then in the mail, came a letter which changed all that. It simply read as follows:

Dear Mr. Woods,

Please accept my renewal to your very excellent magazine. I am one of those carry-overs from SUM, that you took under your wing last summer. Because of your magazine, my interest in the TS2068 has really grown, and I just wanted to say "thank you".

Respectfully Yours,
Jim Preston
Gainesville, Florida

Because of this one note, the whole rest of my day had a different outlook. All of those nights I went without sleep to make a deadline, somehow seemed worthwhile. If I could renew the interest in a powerful little orphan computer, then my original goals for TDM had been met.

Sadly, a few days ago I learned that Jim Preston had suddenly passed away from a heart attack at the age of 66. He had been a member of the Gainesville Sinclair-Timex Users Group for a couple of years. One of the members told me that Jim was "a heck of a guy. Had a great sense of humor...and would do just about anything for you. He was that kind of person".

Although, I never met him personally, I was touched by that one note that came at just the right time. In this respect, I would like to dedicate our March/April 1987 issue of TIME DESIGNS to the memory of Jim Preston.

I think you would have really liked this issue Jim.

Sincerely,
Tim Woods
Managing Editor
Time Designs Magazine Co.

Special Information for TDM Subscribers

WILL YOU BE MOVING SOON? Or even if you change to a post office box, please let our office know well in advance. We have found that the U.S. Postal Service will not reliably forward third class mail (like TDM) even if there was only a small change in the address (like an apartment number for example). To ensure that no issues will be lost, notify us a soon as you know your new address.

WHEN TO RENEW? To determine what your expiration date is, read the information in the upper right-hand corner of your shipping label (located on the front cover of this magazine). For an example: "Mar/87" means that the March/April '87 issue will be the last one you will receive until you renew your subscription again. An early renewal is appreciated. We also send one reminder notice in case you forget. You can also use the form on page 43 to renew your subscription.

TIME DESIGNS MAGAZINE

FOR ALL TIMEX AND SINCLAIR COMPUTERS

TIME DESIGNS MAGAZINE CO.
29722 Hult Rd. • Colton, Oregon 97017
(503) 824-2658
CompuServe ID# 71350,3230

MARCH/APRIL '87
Vol. 3 No. 3

Editor: Tim Woods

Assistant Editor: Stephanie Woods

Editorial Assistant/Production: D.L. Woods

Photography:
(unless otherwise noted): Thomas Judd
Printing by; Toad'l Litho Printing and Comp., Oregon City, Oregon 97045

SUBSCRIPTIONS: $15 a year for six issues (US funds only). No extra charge to Canadian subscribers. All other countries please write for information on air mail rates.

CUSTOMER SERVICE: Customer satisfaction is our goal. For subscription service problems please write or call TIME DESIGNS.

CHANGE OF ADDRESS: Write or call to prevent delay of sevice.

LETTERS

Editor: To "recap" a letter that was printed in the last (Jan/Feb '87) issue of TDM, from Vince Stimmel; help was requested on saving auto-run cassette programs to the A&J Microdrive. Mr. Stimmel reported that he received a number of responses and a phone call before his copy of the magazine arrived in the mail. Here are two responses that were sent in to us directly...

Since we too have recently invested in the "poor folks" mass storage from A&J, we have a feeling of kinship. If you have the customized version of MSCRIPT, the "catalog" function can be a help in getting self-starting and machine code programs onto wafer. Load MSCRIPT; at the HOME menu, set "use" to cassette (just hit the "U" key). Then hit "C" for catalog. Play your program tape, and each program should be listed on the screen, along with memory addresses for all code or the auto-start line number in Basic programs. Now that you have some more information, see if you can break into the Basic and save it to wafer. Then save the indicated code at the indicated address and length.

Branson Wilcox
Cawker City, KS

Thank you for publishing my program "Character ANALYSIS" in the JAN/FEB '87 issue. I hope your readers will enjoy and find some use for the program. I will be happy to answer any of their questions regarding it. I am also enclosing a subroutine for SAVING to A&J and/or tape, to answer the question your reader, Vince Stimmel of Hendersonville, NC, asks. It includes error trapping, verifying, and autostart.

William C. Andrews, M.D.
San Anselmo, CA

TO SAVE GOTO 9000

```
9000 ON ERR RESET : BORDER 6:
 PAPER 6: CLS : PRINT AT 5,12; P
APER 1; INK 9; BRIGHT 1;" SAVE?
";AT 10,7; PAPER 2;" 1 "; PAPER
6; BRIGHT 0;" ON MICROWAFER ";AT
 12,13;" OR ";AT 14,7; PAPER 2;
BRIGHT 1;" 2 "; PAPER 6; BRIGHT
0;" TAPE CASSETTE "; PAUSE 0
9010 LET Z$=INKEY$: IF Z$="2" TH
EN  GO TO 9040
9020 BORDER 1: PAPER 1: CLS : PR
INT AT 10,10; PAPER 2; INK 9; FL
ASH 1;" RECORDING "
9030 SAVE "@1,TITLE" LINE 9100:
GO TO 9100
9040 ON ERR  GO TO 9090: BORDER
0: PAPER 0: CLS : PRINT AT 11,10
; PAPER 2; INK 9; FLASH 1;" RECO
RDING "
9050 SAVE "TITLE" LINE 9100
9060 BORDER 1: PAPER 1: CLS : PR
INT  INK 7;AT 9,2;"REWIND TAPE--
PRESS ANY KEY TO"'''TAB 5;"VERIF
Y OR BREAK TO STOP"
9070 PAUSE 0: CLS : PRINT AT 11,
10; INK 6; FLASH 1;" VERIFYING "
9080 INK 1: VERIFY "": PRINT AT
11,6; PAPER 2; INK 9;" RECORDING
 IS O.K. ": BEEP .5,10: PAUSE 20
0: INK 0: GO TO 9100
9090 CLS : PRINT  PAPER 2; INK 9
; FLASH 1;AT 10,10;" TAPE ERROR
": PAUSE 200
9100 ON ERR RESET : CLS : GO TO
1
```

The number after @ must be in sequence for its position on the wafer.

After the , name of program (7 char. max.).

Number after LINE is the next sequence of program (could be the auto-start).

For tape--no @. Title limited to 10 char. max.

ERROR TRAP

Autostart of program

I am writing to say thank you for doing a great job with TDM. I especially appreciate the Machine Code programs, utilities for the TS 2068. Thank you also for publishing the Source Code for Michael E. Carver's "BASIC2text" (Nov/DEC '86). I really enjoyed that one. I learn so much from labeled, notated source code about the 2068 and the assembler language itself. Thank you for doing this, hope to see more.

Syd Wyncoop's column has been the best presentation of Beginning Z80 Machine Code I've ever seen. Now that the price has come down on Softsync's programs (now available from Zebra Systems): ZEUS ASSEMBLER, ZEUS MONITOR/DISASSEMBLER..well, they are great programs! Here is a tip for the Assembler, that some users may appreciate. The instructions that are included with ZEUS ASSEMBLER do not explain how to get a minus displacement value. For example: you would like to write a program or utility that resides at address 57344 (right at the start of the assembler), and you would like to have your assembled code at address 40000. You would use the ORGinate address 57344. Then to calculate the DISPlacement: 65536 - ORG(address) + assembled code (address) = DISPlacement or 65536 - 57344 + 40000 = DISP 48192. When satisfied with your code, SAVE "name"CODE 40000, length. To test it out, LOAD "name" CODE 57344 or whatever was used for the ORGinate address. Thanks again for a great magazine!

Richard Hurd
Warrenton, OR

Editor: Thanks for your continued support, Richard. I accept both criticism and praise.
Syd Wyncoop replies: I have an easier way for negative displacements in ZEUS. The problem is that my way better lends itself to use in hex. I will usually assemble my code to address #C000 and then add an offset of #2000 to all labels. Using these numbers, the Machine Code would be assembled at 49152 and run from 57344. These numbers are not as strange as they would first appear. They represent page breaks between 8k sections of memory, but this is only obvious when working in hex. The "#" in front of the above numbers is the notation used by ZEUS to denote a hex number. As long as you are running your code from any 256 byte page break, this technique works very well and is easy to use, if you use and understand hexidecimal numbers. Thanks for your comments.

Dear Tim,

You mentioned in the Jan/Feb '87 issue that I would soon complete debugging the TS2068 ROM software. I would have said "debugging is an on going project" and the corrections that are completed are ready for release at any time. I am working on both the TS1000 and TS2068 software. I have the TS1000 (and TS1500) on 16k EPROM. With the price drop on 32k static RAMs we can now have a four chip computer with 32k bytes internal RAM and 8k space for special ROM code.

The latest bugs corrected for the TS2068 include both the HOME ROM and extension ROM software. These corrections allow a BASIC AROS to operate in the advanced video modes. With these corrections the PRINT USR (number) also works in the advanced video modes

Bob Orrfelt
3436 Bay Rd.
Redwood City, CA 94063

I have not been able to figure out how to print a copy of a screen display on my QL Printer. I have tried to interpret the information provided by the QL Users Guide, the QL Printer Manual, and Jan Jones' book "QL SuperBASIC", all to no avail. After spending around $800 for my QL setup, I feel cheated that the capability is not specifically addressed in the manuals, if in fact the capability exists. Must I purchase a program in order to do this, such as a desktop publishing program? Your advice will be much appreciated.

Doug McRoy
Laurel, MD

Mike de Sosa answers: The following short program should solve your problem. This and more useful programs like it will be found in my new book TAKING THE QUANTUM LEAP: THE LAST WORD ON THE SINCLAIR QL, to be published by TIME DESIGNS in April. Always use PAPER 0 (black) when preparing a screen to dump, otherwise you'll wear out your printer cartridge ribbon in a hurry. To make sure you capture all of your screen design, leave about 5 character spaces and 1 line space blank on the top, bottom, and sides of your screen design.

Save the program on your QL-BG (Easel) backup cartridge. Run the program on MDV1. Key and enter "dscreen" to get a suitable window for your design. Make your screen design. Turn your printer on. Key and enter "dump" to copy your screen on your printer.

If you want to SAVE your screen design, use:
SBYTES MDV2_anyname, 131072, 32768
To reload your design, use:
LBYTES MDV2_anyname, 131072

```
1 REMark Gprint_dump
2 a=RESPR(4000)
3 LBYTES mdv1_gprint_prt,a
4 OPEN_NEW #3,mdv1_dcode
5 PRINT #3,a
6 CLOSE #3
7 DEFine PROCedure DUMP
8 OPEN #3,mdv1_dcode
9 INPUT #3,a
10 CLOSE #3
11 CALL a,0,0,0,0,0,0,0,0,0,0
12 END DEFine DUMP
13 DEFine PROCedure DScreen
14 WINDOW 512,256,0,0
15 PAPER 0: INK 7: CLS
16 PAPER #2,0:INK #2,4: CLS #2
17 END DEFine DScreen
```

Build this SUPER SIMPLE MODEM by Joe Williamson

With the popularity of telecommunicating using TS computers, I thought it was about time for a simple, easy to build modem to appear on the scene. Using the circuit shown connected to the Mic jack with the program listed, you too can enjoy telecommunicating over your telephone line. WARNING! There may be rules and regulations governing connections made to telephone lines in your area. Check first.

The Circuit shown can be made from parts that are available locally, and it uses the Mic jack as the I/O port. The transformer matches the impedance of the phone line to the impedance of the computer and provides isolation. The switch allows you to place the circuit "on hook" or "off hook".

The program must be entered exactly as shown, particularly the REM statement which contains the code required for proper operation. The BEEP command is used for the different tones. The program is bare minimum for printing to the screen and keyboard entry. Because of this, you should load in the program first and become familiar with its operation BEFORE attempting the hardware portion. The USR calls are primarily for keeping track of what is printed where on the screen.

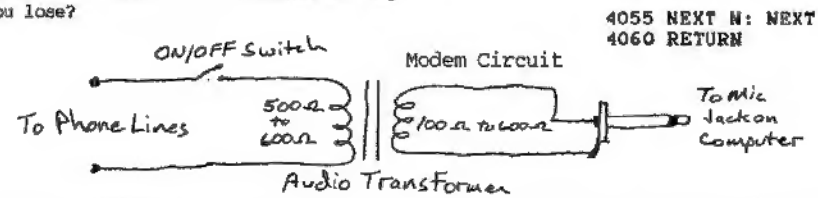
After typing in the program, save it before running so you won't have to type it in again if the program crashes. Once this is done, enter RUN and you should be greeted with the message: "Super Simple Modem Ver. 1.1 Ready". With a flashing cursor here on the next line awaiting keyboard or external entry. Press any key to start.

To go online with some of the different modem services available, turn the switch to the off position and connect to the phone lines and plug into the Mic jack as shown. To dial, use a standard phone and dial up a modem service. As soon as they answer, turn on the switch and hang up the phone. You should be online with who you called.

The simplicity and ease of use of this program will award you great pleasures in use. Try it. Show it to your friends. Enter the world of telecommunicating today. What can you lose?

TS2068 Modem Program

```
  1 REM RTODQ-RHLOKD-LNCDL-UDQ-
O-OQD@CK@OQHKENNKR ()**
  10 BORDER 0: POKE 23693,7: CLS
 : RANDOMIZE USR 26757
  20 LET L=25: LET P=26715: LET
C=IN 244
  30 DIM A$(L+1)
  40 GO SUB 1000: PRINT 'A$
  50 LET P=P+L+1: LET L=4
  55 RANDOMIZE USR 26757: DIM A$
(L+1)
  60 GO SUB 1000: PRINT ''A$
  70 PRINT ' FLASH 1;" "
  80 PAUSE 0: PRINT AT 6,0; FLAS
H 0;" "
  90 LET P=P+L+1: LET L=4: DIM A
$(L+1): GO SUB 1000: LET Q=14:
LET S=.75: LET C=C+4: GO SUB 40
00
 100 LET P=P+L+1: LET L=5: DIM A
$(L+1): GO SUB 1000: LET Q=21:
LET S=.7: LET C=C-2: GO SUB 400
0
 150 BEEP .2,20: BEEP .2,5
 160 BEEP .2,20: BEEP .2,5
 200 PAUSE 0: RANDOMIZE USR 2675
8
 999 STOP
1000 FOR N=0 TO L: LET A$(N+1)=C
HR$ ((PEEK (P+N))+1)
1020 NEXT N
1050 RETURN
3999 STOP
4000 LET A=LEN A$
4025 PRINT AT 21,0; INK 0;a$
4030 FOR f=0 TO 8*a-1: FOR n=0 T
O 7: IF POINT (f,n)=0 THEN  GO
TO 4055
4050 PRINT AT Q-N,F*S; INK C;" "

4055 NEXT N: NEXT F
4060 RETURN
```



NEW PRODUCTS

For TS2068 and SPECTRUM

Artworx

VERSION 1.1

- Pull-Down Menus
- Several Brushes
- Spray
- Auto-Fill
- Zoom
- Undo
- Several Text Fonts
- Cut & Paste Windows
- Auto-Speed Control
- Magnify & Reduce
- Rotate & Mirror
- Full Attribute Control
- Fully Elastic Shapes including Circle, Box, Triangle, Ray and Line
- Fast Ellipse and Arc
- Includes Spectrum & TS2068 Versions
- Supports Microdrives and Kempston Joystick
- Includes GALLERY, the slide show/animator
- 5 Samples of Artwork
- Excellent Manual

Unshackle your creativity with ... ARTWORX!!

ARTWORX V1.1 - $19.95 U.S. plus $3.00 S & H

REVOLUTIONARY NEW BASIC COMPILER...

TIMACHINE

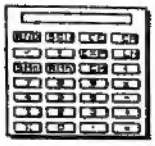
The dream of every BASIC programmer has now been realized!

- TIMACHINE will turn your BASIC into super-fast machine code, running up to 200 times faster!
- Handles floating point operations like SIN, COS, TAN
- Handles all BASIC except I/O
- Includes an excellent manual and 4 demonstration programs
- Compiles up to 27K in seconds
- Includes Spectrum & 2068 versions

Super-Charge your BASIC programs with...*TIMACHINE!*

TIMACHINE — $19.95 U.S. Plus $3.00 S & H

THE WORX!



A unique combination of planning aids, decision tools, and utilities.

- Scheduler/Planner
- Notepad
- Telephone Book
- Programmable Calculator
- Decision Factoring
- Real Time Clock
- Biorhythms
- Perpetual Calendar
- International Time Zones
- Superb Manual
- Includes Spectrum & TS2068 Versions
- Supports Microdrives

A TIMELY ADDITION TO YOUR SOFTWARE LIBRARY...THE WORX!

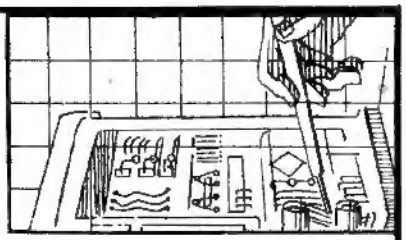
— $19.95 U.S. Plus $3.00 S & H

NovelSoft
A FORMAT FOR THE FUTURE

106 Seventh Street, Toronto, Ontario, Canada M8V 3B4 • TEL. (416)259-8682 • CompuServe 70416,1435

TS Communique

By Joe Williamson



A forum for people having problems with their 1000, 1500 and 2068. If you would like to ask a question, send it to:

TS Comunique
c/o Time Designs Magazine Co.
29722 Hult Road
Colton, OR 97017

I have a Panasonic KX-P1091 printer that I use with *MSCRIPT* on my 2068. I cannot print in italics, pica, elite, or proportional (any multi-byte control codes); I have used various symbols (comma, "", -, etc.), but no luck. Underline, bold, and double width print work well, but nothing else.

This really isn't a "big thing", but it just bothers me that I can't use these functions. Any help you can give me would really be appreciated.

Jack Van Nest
San Diego, CA

Dear Jack,

You must define a code key for each part of the multi-byte control code. For example, to define for italics print (27+52 turns italics on and 27+53 turns italics off), define the code keys as follows in your first line of text:)#0=27,#1=52,#2=53\

When you are ready to insert them into your text, use the function G to add @0@1 to where you want italics to start and @0@2 where you want it to end. -Joe

I am using a Panasonic CT-160 composite color monitor and cartridge software. The color display is excellent from my "main" 2068 computer. The display from my "backup" 2068 is also good while using a TV, but no color from the monitor output.

Was this a common factory problem, and if so, is there a repair that can be made at home without sending the unit through the mail?

John Buckmaster
Maspeth, NY

Dear John,

Yes to both questions. The video from the 2068 is not really up to standards. The color output seems to deviate the most. Your TV probably is a bit more tolerant of these signals and will work with your backup 2068. To help correct the problem, You must open up your 2068. Make sure that the power is off and be careful!

Pop-off the metal cover near the TV output connector and turn the little adjustment inside with a small screwdriver until color appears on your monitor. Be sure and check all the colors to make sure it is stable for all colors. The three adjustments around the speakers also control the display. Adjust VR1 for best jitter free picture. If there is a high-low input impedance switch on your monitor, place it in the "high" position for best operation. VR2 and VR3 control the hue of the colors. VR2 brings out the blues and VR3 brings out the reds. With the tint control on the monitor in the center position, adjust VR2 and VR3 for best hue of yellow, cyan, and magenta. -Joe

Do you know if anyone has a printer interface for the 2068 to connect to a Okidata 10 or 20?

I understand the interface plugs into the printer and is available for the Commodore 64 and IBM PC. Perhaps one could be modified?

I'm told this printer will make color transparencies or with no ribbon, print on thermal paper.

Nathan Willis
Orange, TX

Dear Nathan,

Your best bet would be to go with the IBM type interface and a serial (most likely it's serial) printer interface for the 2068. You should be able to access all of it's capabilities with the driver software for the interface. A serial printer interface should be able to be found from one of TDM's advertisers. If anyone is familiar with this printer being driven by the 2068, drop us a line and let us know. -Joe

I have a 2068 with an Aerco parallel printer interface connected to a Smith-Corona TP-II daisy wheel printer. I am using MSCRIPT with it but unfortunately, I have a "bug" that no one else I know has run into.

It seems that randomly it glitches during a printout. It occasionally forgets to advance the carriage, or it will leave out a letter, add an extra letter, indent when it shouldn't, or not indent when the main text is all indented. The result is that I still frequently find myself doing "cut and paste".

I have no way of determining if it is the program, the interface or the printer. I know it is not the computer because I own two and I have interchanged them with no improvement. I have just recently ordered a new Aerco interface for my second system so I may soon eliminate that item.

Is it possible that my printer has a quirk in it?! Has anyone else out there encountered this sort of frustration with this program? I think that MSCRIPT is great but this "glitch" is most aggravating.

Hel Routt
Clearwater, FL

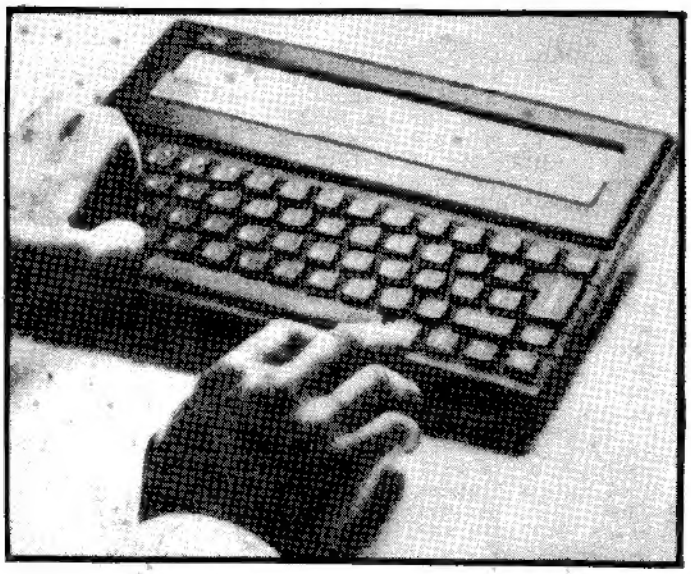
Dear Hel,

Because the "glitch" is so random, it is hard to tell exactly where the problem is coming from but I would suspect the interface and wiring first. Make sure that all connections are clean and tightly fastened. If you have access to an oscilloscope or a logic probe, you can check the data coming out of the cable and work back into the interface making sure that you get good logic levels swinging for less than .5 volts to more than 4.5 volts as data is fed out the cable.

Try flexing the cable while printing to see if the problem exists there. If there is any type of buffer built into the printer, it may take awhile for the results of flexing the cable to appear on the paper. Good luck! -Joe

For Your Sinclair

Two New Sinclair Computers Announced. Sir Clive's Z88 and Spectrum +3.



For several years now, news of a battery-powered "lap computer", code-named PANDORA, was reportedly being developed by Sir Clive Sinclair and his engineering staff. In recent months, Sir Clive himself, talked extensively about the proposed portable microcomputer.

Sharp's Inc. of Mechanicsville, Virginia, reported to TDM in mid-February that the long awaited portable had been unveiled at a British computer show. The new machine, all decked-out in traditional black, signifies a comeback for Clive Sinclair, and is appropriately named the "Z88".

The new Z88 is produced by the Cambridge Computer Company, a subsidary of Sinclair Research. Actual manufacturer of the computer is Thorn EMI. Not surprising, the Z88 will be initially sold by mail order, for about $300 (equivalent U.S. dollars), with an estimated production capacity of 10,000 units per month. At a later date, it will be sold through retail stores.

The portable Z88 does not have some of the features originally proposed by Sinclair, such as use of flat screen television technology, Microdrives, or CP/M. Instead, it uses a new 8 line by 80 column LCD display designed by Epson. The internal processor is a Z80, coupled with on-board 32k RAM. Data and software is stored on battery-backed EPROM cartridges. RAM is expandable to 128k via an optional cartridge. Due to the new wafer scale integration being developed by Sinclair, additional RAM upgrades may be available in the future.

Another feature of the new computer is IBM PC fi compatibility. With an optional software disk for a PC and a cable, the Z88 can upload and download files from an IBM.

The whole unit with four AA batteries weighs less than two pounds, and measures 11.5 inches by 8.5 inches. The full travel keyboard is specially made from silicon, and is said to be totally quiet in use.

Also built-into the Z88 are software programs written by Protechnic of Cambridge, including a word processor a database, a spreadsheet, and some utilities. Further software development from third party houses is highly encouraged by Sinclair.

In further news, Amstrad announced that they will be releasing a 128k Spectrum model that will have a 3" disk drive built in, as opposed to the cassette tape recorder found on the Sinclair Spectrum Plus 2. The 3" disk is the general format of choice in the Amstrad line. The DOS for the Plus 3, will be a customized version of Amsdos. No CP/M compatibility has been announced for the Plus 3.

Desktop Publishing for the TS2068.

Desktop publishing software and hardware packages have taken the personal computer market by storm. By combining a powerful text editor with a graphics development system, the user can produce publications and documents for small business applications, with professional results. Generally, when purchasing one of the commercial desktop packages along with a quality laser printer...the down payment alone could put you into major debt.

Enter Sinclair computers. Already the QL has a budget-priced desktop program called FRONT PAGE. Now the TS2068 joins the ranks with not one program, but two separate offerings.

PIXEL PRINT Desktop Publisher is available for $19.95 ppd., from Lemke Software Development, 2144 White Oak, Wichita, KS 67207. Supports the Tasman, Aerco and A&J printer interfaces.

The TIMEX 2068 DESKTOP PUBLISHER is available for $19.95 + $1.25 for postage and handling, from Charles Stelding, 1415 South Baxter, Tyler, TX 75701.

Largest Timex Sinclair Computer Fest gears up...just weeks away!

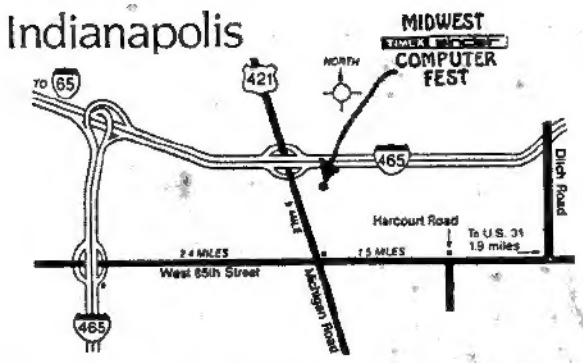
An estimated 1,000 Timex and Sinclair computer users will converge on Indianapolis, Indiana on Saturday May 2nd and Sunday, May 3rd. Attendees will be coming from all over the U.S. and Canada, to participate in the Second Annual Mid-West Timex Sinclair Computer Fest. The event will be housed at the Holiday Inn-North (just off North I-465 at 3850 Depauw Blvd.). The gates open each day at 9a.m., and tickets can be purchased at the door for $6 (individual) or $9 (for a family).

A banquet on Friday evening for dealers and exhibitors will kick off the festivities. A variety of activities are planned for any interest or skill level. Guest speakers will be giving mini-workshops on such subjects as "Graphics and CAD programs", "Using Yo QL", "Machine Code Basics", "Dot Matrix Printers", and "MIDI, Computer Music and Sinclair PC's". Valuable door prizes will be given away hourly, such as new computers, software and books.

Almost every Timex Sinclair vendor will be there, with booths spread over more than 5,000 square feet. Many of these dealers and services have contacted TDM to report that they will be displaying some "never-seen-before" equipment as well as offering substantial Fest price discounts.

Here is an updated list of participating Timex Sinclair vendors and services:

Sharp's Inc.	Novelsoft
Variety Sales	Foote Software
Knighted Computers	The WJDJUP Co.
Zebra Systems Inc.	A+ Computer Response
Time Designs Magazine Co.	Kurt Casby/E. Arthur Brown
C.T.M.	JRC Software
Brice Road Pharmacy	Heath Software
RMG Enterprises	The John Oliger Co.
Quantum Computing	Byte Power
Thomas B Woods/Syncware News	Beaver Computer Products
Vernon Tidwell	Herb Bowers (Abba Soft)
SiriusWare	Haltronics
Curry Computer	Markel Enterprises
Aerco	Sinclair Information Network
Russell Electronics	Lemke Software Development



Booths will also be sponsored by TS User Groups from Indiana, Wisconsin, Ohio, Kentucky, New Jersey, New York, and Michigan.

For further information and details, write or call: Paul Holmgren (Executive Director), 5231 Wilton Wood Ct., Indianapolis, IN 46254, tel.(317) 291-6002; or Frank Davis, 513 E. Main St., Peru, IN 46970, tel.(317) 473-8031.

More New Items...

A TS1000 CLONE??? Yes sir, and several TDM readers have bought one, after seeing advertisements in Computer Shopper and other magazines. The PC8300 was designed and manufactured by Unisonic in Hong Kong during the heyday of the TS1000/ZX81. Although it is similar to the Timex Sinclair, it does offer some improvements like a programmable sound chip (plus a loudspeaker), a joystick port, a monitor output, a chicklet-style keyboard, and an improved tape loading circuit. The PC8300 will accept the TS2040 printer and the TS1016 Rampack...and most (but not all) TS1000 software will load in. The clone is priced to sell at $29.95 + $5.99 for UPS ship, from American Design Components, 62 Joseph St., Moonachie, NJ 07074, tel.(800) 524-0809 or (201) 939-2710.

John Mathewson, 1852 Appleford St., Gloucester, Ontario, Canada K1J 6T4, has developed an external keyboard interface board that plugs into the TS2068's cartridge port, and allows both the computer's keyboard and the external keyboard to operate at the same time. No modifications to the computer required. Price for the interface card is $39.95 (U.S.); for the interface card plus an external keyboard system (in wood cabinent) is $69.95; also available is an interface and external keyboard system for the TS1000/ZX81--write for details.

The TS2068 version of SPECTERM 64 is completed and currently available for $30 + $2 for postage, from G&C Computer Products, PO Box 2186, Inglewood, CA 90305, tel.(213) 759-7406.

The Spectrum option for the Timex/Zebra FDD disk system is now available for $60 from Zebra Systems Inc., 78-06 Jamaica Ave., Woodhaven, NY 11421, tel.(718) 296-2385. Call or write for special ordering instructions. Also available for the FDD from Zebra is the "Software Development Technical Manual" for $20.

A new version of the Larken disk drive interface is ready now. The system for the 2068 and Spectrum consists of an interface board for $45 (U.S.), and the LKDOS/EXBAS Cartridge (which contains all of the commands for the system) for $60. The user supplys the floppy disc drive(s). Some new features include a "snap shot" save routine and 10 new Extended Basic commands. A 256k RAM Disk for the TS2068 will be available soon. Write to: Larken Electronics, RR#2 Navan, Ontario, Canada K4B 1H9.

The DISCIPLE is a new disk drive interface for the Spectrum. Along with the disk feature, it also has a "snap shot" save button, a Centronics printer interface, dual joystick ports, and a networking system similar to the Interface One. All of this comes in one small box for £89.95. For further information, write to: Rockfort Products, 81 Church Road, London, England, NW4 4DP.

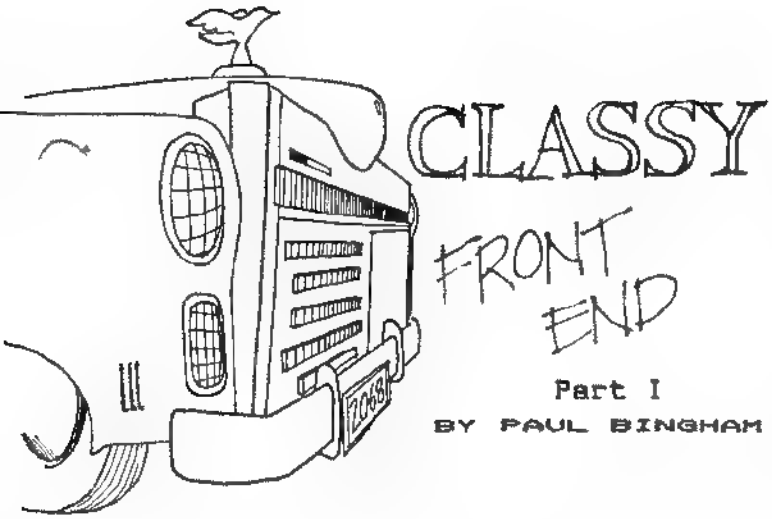
Joe Newman of Variety Sales (325 West Jersey Street #2D, Elizabeth, NJ 07202, tel. 201-527-0535) announced that he would be willing to display software or hardware for any manufacturer or dealer that will not be able to attend the Mid-West Computer Fest. Write or call for details.

Sharp's Inc. (Rt 10 Box 459, Mechanicsville, VA 23111) reported to TDM that a new ROM is available for the QL, that replaces the existing two ROMs inside the computer. Several advantages are possible, including reduced heat build up and crashes. Also, other programs can be added to the new ROM such as I.C.E. or Toolkit II or customized combinations can be ordered. The new ROM is priced at $39.95.

Two new software packages for the QL have been released by Meta Media Productions, 726 West 17th Street, Vancouver, B.C., Canada V5Z 1T9. BOPPERS is a board type game, and FRACTAL is a mathematically based graphics generating program.

The Crocket PAYROLL professional business software package is now available from Kamrec Systems, 51267 E. Village Bldg.17, Apt.205, New Baltimore, MI 48047. Write for information on this and other QL programs.

Version 3.5 of Qflash's RAM-disk and toolkit software will be upgraded shortly to Version 4. It gives RESET128 and COMPARE as SuperBASIC extensions, and other features. The software can be supplied on Microdrive cartridges, an EPROM, or an EPROM for the Sandy SuperQBoard. For prices and further information, write to: Uwe Fischer, Post Box 102121, D-2000 Hamburg 1.



*****Let me start by easing your mind a bit. Just because this is called "Part One", don't assume that you will have to wait for Part Two for the other half of a program or the other half of a discussion! I can't stand waiting for such things and I don't expect any 2068 user reading this to wait either. YOU WILL HAVE A STAND-ALONE PROGRAM WHEN WE ARE THROUGH WITH PART ONE...so read on!

Figure 1: Classy Front End Font

To many a "FRONT END" may be a new term, to others it may already be an old and worn out computer industry buzz word. A FRONT END has come rather loosely to mean any user-friendly working environment for a machine, and is typically replete with fancy fonts, icons & windows. The Front End program is loaded first (or exists in the machine as firmware) and other programs can then be run and manipulated within the new environment.

Epson a few years ago introduced a new computer with such a program, but it ate up memory, slowed down functions to a crawl, and the "many features" got tiring to more experienced users. As is usually the case, others learned from these mistakes and now we have computers like Apple's MAC which owe all of their appeal to their FRONT ENDS.

Although I use some of these other machines, I believe my favorite is still my 2068! Its design under the hood is elegant; simple, uncommon speed and accuracy, and so easily adaptable! One thing I never had a passion for is the character set (or font) that comes with the 2068. But the problems with the font are also found (and much worse) on the C-64 to IBM's PC! Let's look at a few examples to see just what I mean.

Type in the words "big", "abode" and "pound". Do the letters look uneven...sort of like my first-grader wrote them? The fact is they ARE uneven. In order for some letters to have tails (descenders and ascenders), compromises are made. The rounded portions of the letters no longer line up! This same thing is evident in capitals as well. Type in the word "BEEF". Note that all of the horizontals in the middle line up nicely. Now type in "PHRASE". Few of these line up and the look is disjointed.

Now type in "5010". Does the "1" look too small? Try "BRIEF". Does the "I" look too small as well? The lower case makes this even clearer. Type in "whimsical" and "militiaman". Do some of the letters seem to far apart and others too cramped? All of these problems and similar ones on other machines can all be traced to the fact that each character is forced to be in an eight by eight pixel grid. The "i" will never be as wide as the "m" (although the computer designer has attempted it!).

So what to do...it would be of little use to redo the 2068 font using the UDGs. They are 8X8 grids as well. Some computer models have 5X9, 6X8, or other grid sizes (like the QL) to try to compensate. But the only true way to solve the problem is to let an "i" be an "i" and an "m" be an "m". This is the way typesetters do it and it is, in fact, the way the MACINTOSH does it. The question is: "Can the 2068 do it?". The answer is: "Of course!".

The font I have designed we can call NEW 2068 MEDIUM. It is close to ITC Souvenir Medium with a flavor of Clarendon & Helvetica Condensed for all you typesetters. It still has an eight pixel height but the width varies from eight down to three pixels. I have redefined 98 characters, including graphics which will help us with icons, headers and windows later on. If you don't like some of my characters, I won't be offended if you alter them. After all, that's what I'm doing to Uncle Clive's set! If you have a revelation, send it to me, I might want to use it myself!

To print the new font, we cannot easily use the PRINT command. But then the 2068 has at least three ways to put things on the screen. We will use the PLOT routines as they address the screen by pixels and coordinates can be altered easily to accomodate the new three to eight widths. Listing 1 is a program in BASIC and is simple to alter. The program reads the coordinates from lines of BASIC, so can be used along with any programs you care to write. It does not overwrite or interfere with normal 2068 printing, so both can be used at once if you wish!

Continued Next Page.

In Part Two we will look at the machine code version of this program which is faster and can reside most anywhere in memory. For those who have been enjoying the excellent Z80 code lessons in TDM by Syd Wyncoop for the past year, this will be a treat. Most of the instructions have been covered in his articles already!

Although Listing 1 looks incredibly long, it is oh so simple! Line 1000 is the main loop which reads a line of standard 2068 text. It will jump to line 33 to 201, depending on which character code the loop finds next. Simple, huh? If you wish to find which program line draws which character, just look up the character code number...they are listed in the 2068 manual starting with page 239. The rest of the program is housecleaning like keeping track of the end of the screen.

In Figure 1 are sketches of each of the 98 characters in the new font. Under each is the character code and a number showing the pixel width. Most are the same symbol as the original set, four have changed greatly, five were not symbols before but are now. Let's now go over the changes. Number 96 was the famous pound symbol but is now a square icon throwing a shadow. Number 123 (a bracket) is now an inverted underline symbol. Number 124 a graphic symbol on British machines and the STICK command on the 2068 is now a bar symbol (we come full circle!). Number 125 was a bracket and is now a series of lines. Number 126 also a graphic symbol on British machines and the FREE command on the 2068 is now a star. Number 127 was a seldom used copyright symbol and is n the cent symbol. Number 199 was the "less than/equa symbol and is now a small black square. Number 200 was the "greater than/equal" symbol but is now a special code for "Kerning". Typesetters use this to fit letters like "L" and "Y" closer together than they would normally be. Number 184 was the LN symbol but we will use it as a special code to jump to the next line. Number 191 was IN but we will use this as a code to indent twenty spaces. Last is number 201 which was a "does not equal" symbol and is now a hatched line symbol. The usefulness of some of these will become apparent in later discussions, but are fun to experiment with now.

Experimenting will show that even though all the new font's characters are the same height as the original Sinclair set, the spaces have been eliminated. A line of mixed text can now contain sixty or more characters per line! Try out the new symbols and the kerning code. Some possiblilities are shown in Figures 2 and 3. If the listing is beyond your patience to type in, send $4 and I'll send you a copy on tape. Send ideas and any questions you have, too, if you like! My address is: Paul Bingham, POB 2034, Mesa, AZ 85204. See you next time!

Figure 2

```
>>£>>>>>>> THE 2068 COMPUTER CAN
  >>>>>>>£>LN LN IN <= run thou
sands of programsLN IN <= be ine
xpensively expanded£LN IN <= perf
orm many math functionsLN IN <=
entertain youLN IN <= educate yo
u
```

THE 2068 COMPUTER CAN:

- run thousands of programs
- be inexpensively expanded
- perform many math functions
- entertain you
- educate you

Figure 3

```
 ! "#$%& ()*+,-./0123456789 ;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
£abcdefghijklmnopqrstuvwxyz{ STI
CK ~ FREE ©<=<>>=
```

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]↑_£abcdefghijklmnopqrstuvwxyz{|}
~©<=2
```

Listing 1

TS2068

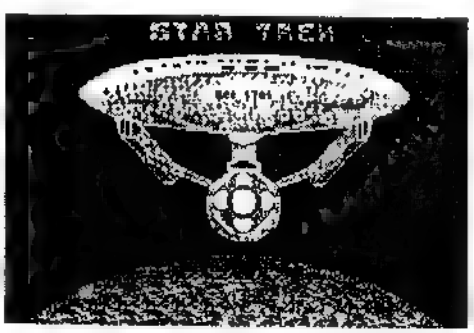
```
  1 REM CLASSY FRONT END Part 1

 10 DIM n$(200): GO TO 1000
 32 GO TO 1130
 33 PLOT x+2,y+1: DRAW 1,0: PLO
T x+2,y+3: DRAW 0,3: DRAW 1,0: D
RAW 0,-3: GO TO 1140
 34 PLOT x+2,y+7: DRAW 0,-2: PL
OT x+5,y+5: DRAW 0,2: GO TO 1170
 35 PLOT x+1,y: DRAW 0,7: PLOT
x+4,y: DRAW 0,7: PLOT x,y+2: DRA
W 5,0: PLOT x,y+6: DRAW 5,0: GO
TO 1160
 36 PLOT x+3,y: DRAW 0,7: PLOT
x+1,y+2: DRAW 1,0: DRAW 0,-1: DR
AW 2,0: PLOT x+4,y+4: DRAW -2,0:
 DRAW 0,2: DRAW 2,0: PLOT x+5,y+
2: DRAW 0,1: PLOT x+1,y+5: GO TO
 1160
 37 PLOT x+3,y+5: DRAW -1,-1: D
RAW -1,0: DRAW 0,1: DRAW 1,1: DR
AW 4,0: DRAW 0,-1: DRAW -1,0: DR
AW 0,-2: DRAW 1,0: DRAW 0,-1: DR
AW -1,-1: PLOT x+3,y+1: DRAW 0,1
: DRAW 1,0: DRAW 0,1: GO TO 1170
 38 PLOT x+4,y+1: DRAW -1,-1: D
RAW 0,1: DRAW -1,0: DRAW 0,4: DR
AW 2,0: PLOT x+1,y+2: PLOT x+3,y
+6: DRAW -2,-2: DRAW 2,-1: DRAW
1,0: GO TO 1150
 39 PLOT x+3,y+7: DRAW 0,-1: DR
AW -1,-1: GO TO 1130
 40 PLOT x+3,y: PLOT x+3,y+7: P
LOT x+2,y+1: DRAW 0,5: PLOT x+1,
y+2: DRAW 0,3: GO TO 1140
 41 PLOT x+1,y: PLOT x+1,y+7: P
LOT x+2,y+1: DRAW 0,5: PLOT x+3,
y+2: DRAW 0,3: GO TO 1140
 42 PLOT x+1,y+4: DRAW 1,1: DRA
W -1,1: PLOT x+3,y+3: DRAW 0,4:
PLOT x+5,y+4: DRAW -1,1: DRAW 1,
1: GO TO 1160
 43 PLOT x+3,y+1: DRAW 0,4: PLO
T x+1,y+3: DRAW 4,0: GO TO 1160
 44 PLOT x+2,y: PLOT x+3,y+1: D
RAW 0,2: DRAW -1,0: GO TO 1140
 45 PLOT x+2,y+3: DRAW 3,0: DRA
W 0,1: DRAW -3,0: GO TO 1170
 46 PLOT x+2,y+1: DRAW 1,0: DRA
W 0,1: DRAW -1,0: GO TO 1150
 47 PLOT x+2,y+1: DRAW 0,1: PLO
T x+3,y+3: DRAW 0,1: PLOT x+4,y+
5: DRAW 0,1: GO TO 1150
 48 PLOT x+1,y+2: DRAW 0,3: PLO
T x+4,y+1: DRAW -2,0: DRAW 0,5:
DRAW 2,0: PLOT x+5,y+2: DRAW 0,3
: PLOT x+3,y+3: PLOT x+4,y+4: GO
TO 1160
 49 GO SUB 1100: DRAW 2,0: DRAW
-1,0: DRAW 0,5: PLOT x+1,y+5: G
O TO 1140
 50 GO SUB 1100: DRAW 0,1: DRAW
1,0: DRAW 0,-1: DRAW 2,0: PLOT
x+1,y+5: PLOT x+2,y+6: DRAW 1,0
: PLOT x+4,y+4: DRAW 0,1: PLOT x+
3,y+3: GO TO 1150
 51 PLOT x+1,y+2: PLOT x+2,y+1:
DRAW 2,0: DRAW 0,5: DRAW -3,0:
PLOT x+5,y+2: DRAW 0,1: PLOT x+5
,y+6: PLOT x+3,y+4: GO TO 1160
 52 PLOT x+3,y+1: DRAW 0,5: DRA
W -1,0: DRAW 0,-1: PLOT x+4,y+3:
DRAW -3,0: DRAW 0,2: GO TO 1150
 53 PLOT x+1,y+2: PLOT x+4,y+6:
DRAW -3,0: DRAW 0,-2: DRAW 2,0:
PLOT x+2,y+5: PLOT x+4,y+2: DRA
W 0,1: PLOT x+2,y+1: DRAW 1,0: G
O TO 1150
 54 PLOT x+4,y+1: DRAW -2,0: DR
AW 0,1: DRAW -1,0: DRAW 0,1: DRA
W 1,0: DRAW 0,2: PLOT x+4,y+4: D
RAW -3,0: PLOT x+5,y+2: DRAW 0,1
: PLOT x+3,y+6: DRAW 1,0: GO TO
1160
 55 PLOT x+1,y+1: DRAW 0,2: DRA
W 1,0: DRAW 0,1: DRAW 1,0: DRAW
0,2: DRAW -3,0: GO TO 1140
 56 PLOT x+4,y+1: DRAW -2,0: DR
AW 0,1: DRAW -1,0: DRAW 0,1: DRA
W 1,0: DRAW 0,3: DRAW 2,0: PLOT
x+1,y+5: PLOT x+5,y+5: PLOT x+3,
y+4: DRAW 1,0: PLOT x+5,y+2: DRA
W 0,1: GO TO 1160
 57 PLOT x+5,y+2: DRAW 0,1: DRA
W -3,0: DRAW 0,3: DRAW 2,0: PLOT
 x+1,y+4: DRAW 0,1: PLOT x+2,y+1
: DRAW 2,0: PLOT x+5,y+4: DRAW 0
,1: GO TO 1160
 58 PLOT x+2,y+4: DRAW 1,0: PLO
T x+3,y+2: DRAW -1,0: GO TO 1140
 59 PLOT x+2,y+4: DRAW 1,0: PLO
T x+2,y: PLOT x+3,y+1: DRAW 0,1:
 DRAW -1,0: GO TO 1140
 60 PLOT x+1,y+3: DRAW 3,-3: PL
OT x+1,y+4: DRAW 3,3: PLOT x+3,y
+2: DRAW -1,1: DRAW 0,1: DRAW 1,
1: GO TO 1150
 61 PLOT x,y+2: DRAW 0,2: PLOT
x,y+4: DRAW 0,2: GO TO 1130
 62 PLOT x+4,y+3: DRAW -3,-3: P
LOT x+4,y+4: DRAW -3,3: PLOT x+2
,y+2: DRAW 1,1: DRAW 0,1: DRAW -
1,1: GO TO 1150
 63 PLOT x+1,y+5: PLOT x+3,y+1
DRAW 1,0: PLOT x+2,y+6: DRAW 2,
0: DRAW 0,-1: DRAW 1,0: DRAW 0,-
1: DRAW -1,0: DRAW 0,-1: DRAW -1
,0: GO TO 1160
 64 PLOT x+3,y+2: DRAW -1,0: DR
AW 0,1: PLOT x+1,y+5: PLOT x+3,y
+4: DRAW 1,0: DRAW 0,-3: DRAW 1,
0: PLOT x+6,y+2: DRAW 0,3: PLOT
x+2,y+6: DRAW 3,0: GO TO 1170
 65 GO SUB 1100: DRAW 0,4: PLOT
 x+2,y+1: DRAW 0,5: DRAW 2,0: PL
OT x+5,y+1: DRAW 0,4: PLOT x+5,y
+3: DRAW -2,0: GO TO 1160
 66 GO SUB 1100: DRAW 0,5: PLOT
 x+4,y+1: DRAW -2,0: DRAW 0,5: D
RAW 2,0: PLOT x+5,y+5: PLOT x+3,
y+4: DRAW 1,0: PLOT x+5,y+2: DRA
W 0,1: GO TO 1150
 67 PLOT x+1,y+2: DRAW 0,3:
T x+4,y+1: DRAW -2,0: DRAW 0,
DRAW 2,0: PLOT x+5,y+2: PLOT x+5
,y+5: GO TO 1160
 68 GO SUB 1100: DRAW 0,5: PLOT
 x+4,y+1: DRAW -2,0: DRAW 0,5: D
RAW 2,0: PLOT x+5,y+2: DRAW 0,3:
 GO TO 1150
```

CLASSY FRONT END

```
 69 PLOT x+5,y+1: DRAW -4,0: DR
AW 0,5: DRAW 4,0: DRAW -3,0: DRA
W 0,-2: DRAW 2,0: DRAW -2,0: DRA
W 0,-2: GO TO 1160
 70 PLOT x+5,y+6: DRAW -4,0: DR
AW 0,-5: DRAW 1,0: DRAW 0,3: DRA
W 2,0: PLOT x+2,y+5: GO TO 1150
 71 PLOT x+4,y+3: DRAW 1,0: DRA
W 0,-2: DRAW -3,0: DRAW 0,5: DRA
W 2,0: PLOT x+1,y+2: DRAW 0,3: P
LOT x+5,y+5: GO TO 1160
 72 GO SUB 1100: DRAW 0,5: DRAW
 1,0: DRAW 0,-5: DRAW 0,3: DRAW
 3,0: DRAW 0,2: DRAW 0,-5: GO TO
 1160
 73 GO SUB 1100: DRAW 2,0: DRAW
 0,5: DRAW -2,0: DRAW 3,0: DRAW
 -2,0: DRAW 0,-5: DRAW 2,0: GO TO
 1150
 74 PLOT x+2,y+1: DRAW 2,0: DRA
W 0,5: DRAW 1,0: DRAW 0,-4: PLOT
 x+1,y+2: DRAW 0,1: GO TO 1160
 75 GO SUB 1100: DRAW 0,5: DRAW
 1,0: DRAW 0,-5: PLOT x+3,y+3: D
RAW 0,1: PLOT x+4,y+2: PLOT x+4,
y+5: PLOT x+5,y+1: PLOT x+5,y+6:
 GO TO 1160
 76 GO SUB 1100: DRAW 0,5: DRAW
 1,0: DRAW 0,-5: DRAW 3,0: GO TO
 1160
 77 GO SUB 1100: DRAW 0,5: DRAW
 1,0: DRAW 0,-5: PLOT x+3,y+5: P
LOT x+4,y+4: PLOT x+5,y+5: DRAW
 1,0: DRAW 0,1: DRAW 0,-5: GO TO
 1170
 78 GO SUB 1100: DRAW 0,5: DRAW
 1,0: DRAW 0,-5: PLOT x+3,y+5: D
RAW 0,-1: PLOT x+4,y+3: DRAW 0,-
1: PLOT x+5,y+1: DRAW 0,5: GO TO
 1160
 79 PLOT x+1,y+2: DRAW 0,3: PLO
T x+4,y+1: DRAW -2,0: DRAW 0,5:
DRAW 2,0: PLOT x+5,y+2: DRAW 0,3
: GO TO 1160
 80 GO SUB 1100: DRAW 0,5: DRAW
 3,0: DRAW -2,0: DRAW 0,-5: DRAW
 0,2: DRAW 2,0: PLOT x+5,y+4: DR
AW 0,1: GO TO 1160
 81 PLOT x+3,y+3: PLOT x+1,y+2:
DRAW 0,3: PLOT x+4,y+2: DRAW 0,
-1: DRAW -2,0: DRAW 0,5: DRAW 2,
0: PLOT x+5,y: DRAW 0,5: GO TO 1
160
 82 GO SUB 1100: DRAW 0,5: DRAW
 0,0: DRAW -2,0: DRAW 0,-5: DRAW
 0,2: DRAW 2,0: DRAW 0,-1: DRAW
 1,0: DRAW 0,-1: PLOT x+5,y+4: DR
AW 0,1: GO TO 1160
 83 PLOT x+1,y+2: DRAW 1,0: DRA
W 0,-1: DRAW 2,0: PLOT x+4,y+4:
DRAW -2,0: DRAW 0,2: DRAW 2,0: P
LOT x+5,y+2: DRAW 0,1: PLOT x+1,
y+5: GO TO 1160
 84 PLOT x+1,y+6: DRAW 2,0: DRA
W 0,-5: DRAW 1,0: DRAW 0,5: DRAW
 2,0: GO TO 1160
 85 PLOT x+1,y+2: DRAW 0,4: DRA
W 1,0: DRAW 0,-5: DRAW 2,0: PLOT
 x+5,y+2: DRAW 0,4: GO TO 1160
 86 PLOT x+1,y+5: DRAW 0,1: DRA
W 1,0: DRAW 0,-3: DRAW 1,0: DRAW
 0,1: DRAW 0,-3: DRAW 1,0: DRAW
 0,1: PLOT x+5,y+3: DRAW 0,1: PLO
T x+6,y+5: DRAW 0,1: GO TO 1160
 87 PLOT x+1,y+2: DRAW 0,4: DRA
W 1,0: DRAW 0,-5: DRAW 1,0: DRAW
 0,1: DRAW 1,0: DRAW 0,2: DRAW 0
,-2: DRAW 2,0: DRAW 0,4: PLOT x+
5,y+1: GO TO 1170
 88 GO SUB 1100: DRAW 1,0: DRAW
 0,1: DRAW 1,0: DRAW 0,1: DRAW 1
,0: DRAW 0,1: DRAW -1,0: DRAW 0,
1: DRAW -1,0: DRAW 0,1: DRAW -1,
0: PLOT x+5,y+2: PLOT x+5,y+5: P
LOT x+6,y+1: PLOT x+6,y+6: GO TO
 1170
 89 PLOT x+4,y+4: DRAW 0,-3: DR
AW -1,0: DRAW 0,3: DRAW -1,0: DR
AW 0,1: DRAW -1,0: DRAW 0,1: PLO
T x+6,y+6: PLOT x+5,y+5: GO TO 1
160
 90 PLOT x+1,y+6: DRAW 4,0: DRA
W 0,-1: DRAW -1,0: DRAW 0,-1: DR
AW -1,0: DRAW 0,-1: DRAW -1,0: D
RAW 0,-1: DRAW -1,0: DRAW 0,-1:
DRAW 4,0: GO TO 1160
 91 PLOT x+2,y: DRAW -1,0: DRAW
 7: DRAW 1,0: GO TO 1130
 92 PLOT x+4,y+1: DRAW 0,1: PLO
T x+3,y+3: DRAW 0,1: PLOT x+2,y+
5: DRAW 0,1: GO TO 1150
 93 PLOT x+1,y: DRAW 1,0: DRAW
0,7: DRAW -1,0: GO TO 1130
 94 PLOT x+1,y+4: DRAW 2,2: DRA
W 2,-2: DRAW -2,2: DRAW 0,-5: GO
 TO 1160
 95 PLOT x,y: DRAW 7,0: GO TO 1
1160
```

```
 95 PLOT x+6,y: DRAW -5,0: DRAW
0,5: DRAW 1,0: DRAW 0,1: DRAW 5
,0: DRAW 0,-5: DRAW -5,0: DRAW 0
,3: GO TO 1180
 97 PLOT x+4,y+4: DRAW 0,-3: DR
AW -2,0: DRAW 0,3: DRAW 1,0: PLO
T x+1,y+2: DRAW 0,1: GO TO 1150
 98 GO SUB 1100: DRAW 0,5: DRAW
 0,-2: DRAW 2,0: DRAW 0,-3: DRAW
 -1,0: PLOT x+4,y+2: DRAW 0,1: G
O TO 1150
 99 PLOT x+1,y+2: DRAW 0,1: PLO
T x+4,y+1: DRAW -2,0: DRAW 0,3:
DRAW 2,0: GO TO 1150
 100 PLOT x+4,y+6: DRAW 0,-5: DR
AW -2,0: DRAW 0,3: DRAW 1,0: PLO
T x+1,y+2: DRAW 0,1: GO TO 1150
 101 PLOT x+4,y+3: DRAW 0,1: DRA
W -2,0: DRAW 0,-3: DRAW 2,0: DRA
W -1,0: DRAW 0,1: DRAW -2,0: DRA
W 0,1: GO TO 1150
 102 PLOT x+1,y+4: DRAW 2,0: PLO
T x+2,y+1: DRAW 0,5: PLOT x+3,y+
6: GO TO 1140
 103 PLOT x+4,y: DRAW 0,4: DRAW
-2,0: DRAW 0,-3: DRAW 1,0: PLOT
x+1,y+3: DRAW 0,-1: PLOT x+4,y
DRAW -3,0: GO TO 1150
 104 GO SUB 1100: DRAW 0,5: PLOT
 x+2,y+4: DRAW 1,0: PLOT x+4,y+3
: DRAW 0,-2: GO TO 1150
 105 GO SUB 1100: DRAW 2,0: DRAW
 -1,0: DRAW 0,3: DRAW -1,0: PLOT
 x+2,y+6: GO TO 1140
 106 PLOT x,y: DRAW 1,0: PLOT x+
2,y+1: DRAW 0,3: PLOT x+2,y+6: G
O TO 1130
 107 GO SUB 1100: DRAW 0,5: PLOT
 x+2,y+3: DRAW 1,0: DRAW 0,-1: P
LOT x+4,y+1: PLOT x+4,y+4: GO TO
 1150
 108 GO SUB 1100: DRAW 2,0: DRAW
 -1,0: DRAW 0,5: PLOT x+1,y+6: G
O TO 1140
 109 GO SUB 1100: DRAW 0,3: PLOT
 x+2,y+1: DRAW 0,3: DRAW 3,0: PL
OT x+4,y+1: DRAW 0,2: PLOT x+6,y
+1: DRAW 0,2: GO TO 1170
 110 GO SUB 1100: DRAW 0,3: PLOT
 x+2,y+1: DRAW 0,3: DRAW 1,0: PL
OT x+4,y+1: DRAW 0,2: GO TO 1150
 111 PLOT x+3,y+1: DRAW -1,0: DR
AW 0,3: DRAW 1,0: PLOT x+1,y+2:
DRAW 0,1: PLOT x+4,y+2: DRAW 0,1
: GO TO 1150
 112 PLOT x+1,y: DRAW 0,4: DRAW
2,0: DRAW 0,-3: DRAW -1,0: PLOT
x+4,y+3: DRAW 0,-1: GO TO 1150
 113 PLOT x+4,y: DRAW 0,4: DRAW
-2,0: DRAW 0,-3: DRAW 1,0: PLOT
x+1,y+3: DRAW 0,-1: GO TO 1150
 114 GO SUB 1100: DRAW 0,3: PLOT
 x+2,y+1: DRAW 0,2: PLOT x+3,y+4
: DRAW 1,0: GO TO 1150
 115 GO SUB 1100: DRAW 2,0: DRAW
 0,2: DRAW -2,0: PLOT x+4,y+2: P
LOT x+4,y+4: DRAW -2,0: GO TO 11
50
 116 PLOT x+2,y+1: DRAW 0,5: PLO
T x+1,y+4: DRAW 2,0: PLOT x+3,y+
1: GO TO 1140
 117 PLOT x+1,y+2: DRAW 0,2: PLO
T x+2,y+4: DRAW 0,-3: DRAW 2,0:
DRAW 0,3: GO TO 1150
 118 PLOT x+1,y+4: DRAW 0,-1: DR
AW 1,0: DRAW 0,-1: DRAW 1,0: DRA
W 0,-1: PLOT x+4,y+3: DRAW 0,1
GO TO 1150
 119 PLOT x+1,y+2: DRAW 0,2: PLO
T x+2,y+4: DRAW 0,-3: DRAW 3,0
DRAW -1,0: DRAW 0,2: PLOT x+6,y+
2: DRAW 0,2: GO TO 1170
 120 GO SUB 1100: PLOT x+1,y+4
PLOT x+4,y+4: PLOT x+4,y+1: PLOT
 x+2,y+2: DRAW 1,0: DRAW 0,1: DR
AW -1,0: GO TO 1150
 121 PLOT x+1,y: DRAW 1,0: PLOT
x+3,y+1: PLOT x+1,y+3: DRAW 0,1:
 DRAW 1,0: DRAW 0,-2: DRAW 2,0
DRAW 0,2: GO TO 1150
 122 GO SUB 1100: DRAW 3,0: PLOT
 x+1,y+4: DRAW 3,0: PLOT x+2,y+2
: PLOT x+3,y+3: GO TO 1150
 123 PLOT x,y+7: DRAW 5,0: GO TO
1160
```

```
 124 PLOT x+1,y: DRAW 0,7: PLOT
x+2,y: DRAW 0,7: GO TO 1140
 125 PLOT x,y: DRAW 5,0: PLOT x,
y+2: DRAW 5,0: PLOT x,y+4: DRAW
5,0: PLOT x,y+6: DRAW 5,0: GO TO
 1160
 126 GO SUB 1100: DRAW 3,3: PLOT
 x+7,y+1: DRAW -2,2: PLOT x+2,y+
1: DRAW 4,4: PLOT x+5,y+3: DRAW
2,2: PLOT x+6,y+1: DRAW -4,4: PL
OT x+4,y+4: DRAW 0,3: PLOT x+3,y
+3: DRAW -2,2: GO TO 1180
 127 PLOT x+1,y+3: DRAW 0,1: PLO
T x+4,y+2: DRAW -2,0: DRAW 0,3
DRAW 2,0: PLOT x+3,y: DRAW 0,2
PLOT x+3,y+5: DRAW 0,2: GO TO 11
50
 184 GO SUB 1300
 191 LET x=x+20: RETURN
 199 PLOT x+1,y+3: DRAW 0,1: PLO
T x+2,y+3: DRAW 0,1: GO TO 1130
 200 LET x=x-1: RETURN
 201 PLOT x,y: DRAW 2,0: PLOT x,
y+1: DRAW 2,2: PLOT x,y+4: DRAW
2,2: PLOT x,y+7: DRAW 2,0: GO TO
 1130
 999 RETURN
1000 INPUT "PHRASE ";n$
1010 INPUT "PRINT AT ";" FLASH 1,
"?";" FLASH 0;",x (to 21).";yy
1020 INPUT "PRINT AT y,";" FLASH
1,"?";" FLASH 0;" (to 31).";xx
1030 PRINT n$
1040 LET x=xx*8: LET y=168-yy*8:
 FOR t=1 TO 200: LET a=CODE n$(t
): GO SUB a: NEXT t
1050 STOP
1100 PLOT x+1,y+1: RETURN
1120 LET x=x+2: IF x>248 THEN GO
 TO 1300
1125 RETURN
1130 LET x=x+3: IF x>248 THEN GO
 TO 1300
1135 RETURN
1140 LET x=x+4: IF x>248 THEN GO
 TO 1300
1145 RETURN
1150 LET x=x+5: IF x>248 THEN GO
 TO 1300
1155 RETURN
1160 LET x=x+6: IF x>248 THEN GO
 TO 1300
1165 RETURN
1170 LET x=x+7: IF x>248 THEN GO
 TO 1300
1175 RETURN
1180 LET x=x+8: IF x>248 THEN GO
 TO 1300
1185 RETURN
1300 LET x=0: LET y=y-8: RETURN
9999 SAVE "CFE"
```



STAR TREK

A Game of strategy

T/S 2064 --7 LEVELS of play--

Will you accept the challenge?

Send $8.00 to:
Michael E. Carver
1016 NE Tillamook
Portland, OR 97212

Based on the Mikebug Public Domain program

GAME

CUBE-IT

By Ralph Hammer

The following program listing is for the Timex Sinclair 2068. CUBE-IT is a game loosely based on the JUMBLE Word-Cube game by Parker Brothers, which was popular about 5 to 10 years ago. Instead of a 5X5 grid, I use a 6X6. The letters are chosen completely at random from the A$ (Line 30) in order of most used. The game also features full instructions on the screen, a two-minute timer, a copy option using the TS2040 printer, and a scoring ledger. CUBE-IT is fun to play for ages 6 to 96. Give it a try!

I will also send the program on a high-quality tape to anyone who does not want to type in the LLISTing, for $4.00. Ralph Hammer, 7 Baer Drive, Las Vegas, NV 89115.

```
  10 REM \CUBE-IT\VERSION 2.1\7-
86\RWH\
  20 CLEAR 64736: POKE 23658,8
LET MUSIC=700: LET PNT=320: LET
THIN=880: GO SUB 800: GO SUB THI
N
  30 LET A$="ETAONRISHDLFCMUGYPW
BVKXJQZ": LET L$="
  50 REM Continue
 100 REM -TITLE PAGE-
 105 BORDER 5: BRIGHT 1: PAPER 5
: INK 0: BRIGHT 0: CLS
 110 PRINT '           ,TAB 21,"
"
 115 PRINT           ",TAB 22,"
 120 PRINT           ",TAB 22,"
 125 PRINT           ",TAB 22,"
 130 PRINT           ",TAB 22,"
 135 PRINT '          ",TAB 21,"

 140 PLOT 64,150: DRAW 0,-30: DR
AW 20,0: PLOT 83,150: DRAW 0,-30
: DRAW 5,0: PLOT 66,150: DRAW 0,
-30: PLOT 83,150: DRAW 0,-30
 145 PLOT 96,156: DRAW 0,-46: DR
AW 20,0: DRAW 0,30: DRAW -18,0:
PLOT 98,156: DRAW 0,-46: PLOT 11
4,120: DRAW 0,30
 150 PLOT 123,135: DRAW 15,0: DR
AW 0,15: DRAW -20,0: DRAW 0,-30:
DRAW 20,0: DRAW 0,5: PLOT 144,1
50: DRAW 0,-15: PLOT 126,150: DR
AW 0,-30: PLOT 144,120: DRAW 0,5
 155 PLOT 208,166: DRAW 0,-46: D
RAW 20,0: DRAW 0,5: PLOT 198,150
: DRAW 22,0: PLOT 210,166: DRAW
0,-46: PLOT 226,120: DRAW 0,5
 160 PRINT '" A FUN-FOR-ALL WORD
SEARCH GAME"
 165 PRINT "By: Ralph Hammer, La
s Vegas, Nev",L$
 175 REM -POETRY-
 180 GO SUB FAT: PRINT ' 'GRAB A
PEN, AND SOME PAPER, IT'S TIME T
O PLAY A WORD SEARCH CAPER"
 185 PRINT "PLAY IT SOLO, E.EN
TWO OR MORE, SEE WHO GETS THE FI
RST HI SCORE."
 190 PRINT "NOW DON'T BE SHY, G
A VERY TARDY.PUSH A KEY FOR A SE
ARCHING PARTY"
 195 GO SUB THIN: GO SUB MUSIC
 200 PAUSE NOT X: BEEP .05,5
 225 REM -MENU-
 230 BORDER 4: BRIGHT 1: PAPER 4
: CLS
 235 PRINT INK 7'TAB 10, PAPER 2
, "CUBE-IT "
 240 INK 0: GO SUB FAT: PRINT TA
B 10; PAPER 6;" M E N U
 245 PRINT AT 7,0 "SELECT 1-5:
 GO SUB THIN: PRINT '' 1. INSTRU
CTIONS";'" 2. HOW TO SCORE";'"CU
E-IT";" 3. PLAY GAME";' 4. M
AKE BACK-UP TAPE";' " 5. QUIT"
 255 PAUSE 0: LET I$=INKEY$: BEE
P .05,5
 260 IF I$="1" THEN GO TO 900
 265 IF I$="2" THEN GO TO 1100
 270 IF I$="3" THEN GO TO 300
 275 IF I$="4" THEN GO TO 1300
 280 IF I$="5" THEN GO TO 1450
 285 GO TO 255
 300 REM -GAME-
 305 BORDER 6: BRIGHT 1: PAPER 6
: CLS
 310 GO SUB 1500: GO SUB THIN
 315 PRINT FLASH 1; AT 12,2; "■■I
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
 320 FOR I=1 TO 10: FOR J=0 TO 1
5: BEEP .003,J: NEXT J: NEXT I
 325 GO SUB 1500: RANDOMIZE: LE
T P=PEEK 23670+256*PEEK 23671
 330 GO SUB 1700: GO SUB THIN
 400 REM -TIMER-
 410 BEEP .2,10: PAPER 2: INK 7
: BRIGHT 1: PRINT AT 18,10,"

 415 LET Y=PEEK 23672: FOR M=0 T
O 1: FOR S=0 TO 59: FOR T=0 TO 9
 420 LET X=PEEK 23672: IF ABS (X
-Y)<=5 THEN GO TO 420
 425 LET Y=X: PRINT AT 18,13;M;"
";S;TAB 17;"";T: NEXT T: BE
EP .005,1
 430 NEXT S: NEXT M: BEEP .3,-5
: BEEP .3,15
 435 PRINT AT 18,13;"2 00 0": BE
EP .3,10: BEEP .3,15
 440 INK 0: PRINT AT 20,10; FLAS
H 1; PAPER 7;" TIME IS UP "
 445 PAPER 6: PAUSE 120
```

```
500 REM -CHOOSE-
505 GO SUB 1500: PRINT AT 9,2,"
<R> RE-DRAW CUBE FOR SCORING"
510 PRINT '"   <C> COPY CUBE TO
PRINTER"''"   <A> ANOTHER GAME"'
"   <M> MAIN MENU"' GO SUB THIN
   PAUSE 0: LET I$=INKEY$: IF
   C" THEN LET S=0: GO TO 530
   IF I$="A" THEN GO TO 300
525 IF I$<>"R" THEN GO TO 225
530 PAPER 4: GO SUB 1500: RANDO
MIZE R: GO SUB 1700: GO SUB 1200
: GO SUB THIN: IF NOT S THEN COP
Y
535 PRINT #0;AT 1,0;"     <C>OPY
   <P>LAY   <M>ENU": PAUSE 0: IF I
NKEY$="C" THEN COPY: GO TO 535
540 IF INKEY$="P" THEN GO TO 30
0
545 GO TO 225
600 REM -FRAME-
605 PLOT 57,150: DRAW 142,0: PL
OT 58,149: DRAW 139,0: PLOT 59,1
48: DRAW 138,0
610 PLOT 57,50: DRAW 142,0: PLO
T 58,51: DRAW 139,0: PLOT 59,52:
DRAW 138,0
615 PLOT 199,150: DRAW 0,-100:
PLOT 198,15■: DRAW 0,-99
620 PLOT 57,150: DRAW 0,-100: P
LOT 58,149: DRAW 0,-99: PLOT 199
,150: DRAW 0,-100: PLOT 198,150:
DRAW 0,-99
625 LET X=57: LET Y=140: LET Z=
0
630 PLOT X,132: DRAW Y,Z: PLOT
X,115: DRAW Y,Z: PLOT X,100: DRA
W Y,Z: PLOT X,84: DRAW Y,Z: PLOT
X,68: DRAW Y,Z
635 LET X=150: LET Y=0: LET Z=-
99
640 PLOT 80,X: DRAW Y,Z: PLOT 1
04,X: DRAW Y,Z: PLOT 128,X: DRAW
Y,Z: PLOT 152,X: DRAW Y,Z: PLOT
176,X: DRAW Y,Z
645 RETURN
700 REM -MUSIC-
705 FOR X=0 TO 100
710 LET F=INT (RND*256)
715 SOUND 0,F,8,15,7,62
720 PAUSE 10: IF INKEY$<>"" THE
N LET X=0: GO TO 730
   NEXT X
   SOUND 7,63,8,0: RETURN
   REM -MC FAT-
805 DATA 17,0,253,213,1,0,3,42,
54,92,36,126,167,31,182,18,35,19
,19,32,246,16,244,225,37,34,54,9
2,201
810 FOR I=64737 TO 64765
815 READ J: POKE I,J: NEXT I: R
ETURN
850 REM -FAT-
855 RANDOMIZE USR 64737: RETURN
880 REM -THIN-
885 POKE 23607,60: RETURN
900 REM -INSTRUCTIONS-
905 BORDER 5: BRIGHT 1: PAPER 5
: CLS
910 GO SUB 1500
915 PRINT INK 0;TAB 9; PAPER 6;
" INSTRUCTIONS "
920 PRINT '''AS YOU START THE G
AME OF CUBE-ITA 6X6 LETTERED GRI
D WILL APPEAR."
925 PRINT '"NEXT, A TWO MIN
UTE TIMER    WILL START TICKING
OFF ELAPSED  TIME."
930 PRINT '"START LOOKING FOR T
WO, THREE,   UP TO SIX (OR MORE)
LETTER WORDSTHAT CAN BE FOUND W
ITHIN THE    GRID."
935 PRINT '"WRITE DOWN ALL THE
WORDS THAT   YOU CAN FIND WITHIN
THE TWO      MINUTE TIME LIMIT."
940 GO SUB THIN: PRINT '''   PRE
SS ANY KEY TO CONTINUE"
945 PAUSE 0: CLS: GO SUB FAT
BEEP .05,5
950 PRINT '"WORDS CAN BE FOUND
IN ANY DIREC-TION, HORIZONTALLY,
VERTICALLY, AND DIAGONALLY."
955 PRINT '"THEY CAN ALSO BE A
COMBINATION  OF ALL THREE DIRECT
IONS."
960 PRINT '"WHEN FORMING A WORD
, A LETTER   MUST TOUCH THE NEXT
LETTER       EITHER AT THE LINE,
OR AT A      CORNER."
965 PRINT '"DO NOT JUMP SQUARES
TO FORM THE WORD.  DO NOT USE A
LETTER TWICE"
970 PRINT '"FOR EXAMPLE "
975 PRINT '"""ABY"" CANNOT BE U
SED TO SPELL   BABY."
980 GO SUB THIN: PRINT "   PRE
SS ANY KEY TO CONTINUE"
985 PAUSE 0: CLS   GO SUB FAT:
BEEP .05,5
990 PRINT '"""FOUND"" WORDS CAN
NOT BE REUSED OR SCORED TWICE."
995 PRINT "IF TWO OR MORE PEOP
LE ARE GOING TO PLAY, IT IS RECO
MMENDED THAT THE FIRST PLAYER TO
REACH 500   POINTS BE DECLARED
THE WINNER."
1000 PRINT '"ALL WORDS, INCLUDIN
G LEGAL NAMESAND ""SLANG"" WORDS
ARE PERMITTED,"
1005 PRINT '"ABBREVIATIONS AND F
OREIGN WORDS ARE NOT ALLOWED."
1010 PRINT '"GOOD LUCK, AND HAVE
FUN !!!"
1020 GO TO 1600
1100 REM -SCORE-
1105 BORDER 5: BRIGHT 1: PAPER 5
1110 GO SUB 1500
1115 PRINT AT 1,19; INK 7; PAPER
2;" I T ";AT 2,9; INK 0; FLASH
1; PAPER 6;" S C O R I N G "
1120 PRINT '" TO SCORE THE GAME
OF ""CUBE-IT""''
1125 PRINT "JUST COUNT THE NUMBE
R OF LETTERSIN EACH WORD."
1130 PRINT '"POINT VALUES ARE DI
STRIBUTED AS FOLLOWS "
1135 GO SUB 1200: GO TO 1600
1200 PRINT PAPER 6;" # OF LETTER
S",TAB 20,"POINT VALUE "
1210 DATA 1,2,3,10,25
1220 RESTORE 1140: FOR I=2 TO 6
: READ J: PRINT TAB 4; I;TAB 14; -
";TAB 26-LEN STR$ J;J: NEXT I
1230 RETURN
1300 REM *TAPE BACK-UP*
1305 BORDER 2: BRIGHT 1: PAPER 2
: CLEAR
1310 SAVE "CUBE" LINE 10: BEEP 2
,25: PRINT INK 7;"Play tape  Pre
ss a key to verify"
1315 PAUSE 0: CLS   VERIFY "CUBE
": BEEP 2,25: RUN
1350 REM -QUIT-
1355 BORDER 1: BRIGHT 1: PAPER 5
: CLS
1360 SOUND 7,62,8,15
1365 FOR I=50 TO 100
1370 SOUND 0,I: PAUSE 3
1375 NEXT I
1380 FOR D=1 TO 5: NEXT D
1385 SOUND 6,6,7,7,8,16,9,16,10,
16;12,56;13,8
1390 PAUSE 90
1395 SOUND 8,0,9,0,10,0
1400 PRINT FLASH 1;AT 11,11," SO
LONG ", FLASH 0: STOP
1500 REM -LOGO-
1510 CLS: INK 7: PRINT TAB 9;
PAPER 2;" C U B E   I T "
1520 GO SUB FAT: PLOT 142,164: D
RAW 6,0: INK 0: RETURN
1600 REM -SUBMENU-
1610 PRINT AT 17,0;L$;"     ": FL
ASH 1;"M", FLASH 0;"ENU",
1620 PRINT TAB 14; FLASH 1;"P"
FLASH 0 "LAY";
1630 PRINT TAB 25; FLASH 1;"Q",
FLASH 0 "UIT" L$
1640 GO SUB THIN: PRINT TAB 7;"S
ELECT M, P, OR Q"
1650 PAUSE 0: LET I$=INKEY$: BEE
P .05,5: IF I$="M" THEN GO TO 22
5
1660 IF I$="P" THEN GO TO 300
1670 IF I$="Q" THEN GO TO 1350
1680 GO TO 1650
1700 REM - PRINT LETTERS-
1710 PRINT '   BORDER 2: LET I=U
SR 22177: FOR I=1 TO 6
1720 PRINT TAB 8;   FOR L=1 TO 6
1730 LET T=INT (RND*26+1): IF T>
19 THEN IF INT (RND*2) THEN LET
T=0
1740 IF T>9 THEN IF INT (RND*2)
THEN LET T=0
1750 IF NOT T THEN GO TO 1730
1760 PRINT A$(T);"  ";
1770 NEXT J: PRINT '': NEXT I
1780 GO SUB 600: RETURN
```

"The Mystery of the Missing 253"

CONCLUSION

By Wes Brzozowski

Where We've Been

It's been wonderful to see how much interest there still is toward TS2068 bank switching. Now it's time to tie up the loose ends we've left, provide a firm direction for your own further study, and close the series up.

For those just tuning in, we've been discussing the ROM code, searching the Technical Manual, and scouring various obscure sources to learn all we can about Timex's original plans for extended bank switching. Timex originally promised 256 banks of 64k each, and delivered a machine that could passably control 3 (really only 2 1/8, for you purists out there).

We've uncovered a method that could logically control the "Missing 253". (We've seen that the ROM software as is, limits itself to only 10 extra banks, but this is a mere technicality.) We then took a basic look at how that method works. We saw how the hardware and software were to have meshed together through only two small subroutines. This is a good system design practice and is a typical example of how hardware should be controlled by a processor. This allows hardware changes to be corrected through minimal software modifications, and will be a key factor in making extended bank switching practical.

Following all this, we endured an excruciatingly detailed description of what the hardware must and must not do. (Alternately, this is what the system must be able to SIMULATE.) We've been going through the EXROM software that initializes the banks, calling attention to many obvious bugs along the way. All of this would be needed for an interested user to design a bank switching system that simply wouldn't crash the TS2068 on power-up.

As a secondary effort, we've looked at an easy way to correct the ROM bugs, and toward the special I/O software that would have dovetailed with bank switching. We've little more than hinted about various blocks of I/O software in the ROM that are never used. We've also seen a little of how the RAM Resident Code would have supported bank switching, and will supply a better description before we re through.

First, Finish the Flippin' Flowcharts

Flowcharts 7 and 8 cover the last of the initialization software for the expansion banks. Both point out major bugs in the routines that initialize RAM banks and renumber the banks. To save text here, I've tried to make these two as self-explanatory as possible.

If you've made up a memory map of the EXROM routines we've covered, you'll see two gaping holes. The smaller of the two is merely a copy of Spectrum code that initializes some of the more mundane system variables. These are put off until after the bank switching is initialized, because programs running in the cartridge slot can take up some extra memory, affecting the values of these variables. However, this routine is of no interest to us, and should be left alone.

The larger hole is filled by an unused routine that a performs a "warm reset" on the SYSCON table. This was once intended to be accessed by certain forms of the RESET command, from BASIC. We'll discuss the RESET command a little more in a moment, but flowcharting this routine should be an excellent exercise for those of you who've followed the series this far. Since the routine is not essential to this subject, we can't cover it further here.

Our detailed discussion of the code has now covered all the useful initialization routines, the lowest-level RAM Resident Code that actually communicates with the hardware, and the intermediate-level RAM Resident Code, which talks to the low-level stuff. (Whew!)

A Promise Fulfilled

A long, long time ago, I said I'd provide a better description of the RAM Resident Code, and the routines that don't relocate properly to high memory. It's time to set things right. The full page table, titled "RAM Resident Code - Routines, Usage, and Notes", is a reasonably complete "cheat sheet" on using the routines. Hopefully it's fairly free of typos and its small print will reproduce well on these pages. This will tell you most everything you need to know in order to use the routines, including how to correct them in low memory. Note that the following locations do NOT relocate properly when the code is in high memory:

FC69,A	FCCE-FCD0	FF04/5
FC6C,D	FCD6-FCD8	FF0F-FF11

Without correcting the EXROM, you'll have to fix these every time the second display file is opened. CAUTION: you also have to change them BACK before closing the second display file, or the relocation to low memory will be messed up.

By the way, the good folks at Timex left an interesting chicken-and-egg situation. It shouldn't be hard to imagine code that contains routines that open and close the second display file, and also routines that use the RAM Resident Code. If they operate independently, it may become necessary to find out where the RAM Res Code is at any given moment, so you'll know whether to CALL the low or the high addresses.

Now, the "standard" Timex way of finding out is to check the system variable VIDMOD, which will be non-zero if the second display file is open, and hence, the RAM Res Code is in high memory. The problem arises when the memory chunk containing VIDMOD is not enabled for the Home Bank; how do you get at the variable? Well, we might first consider one of the RAM Resident Routines, GET_WORD, which can read the contents of any memory location in any bank. That can read the variable.

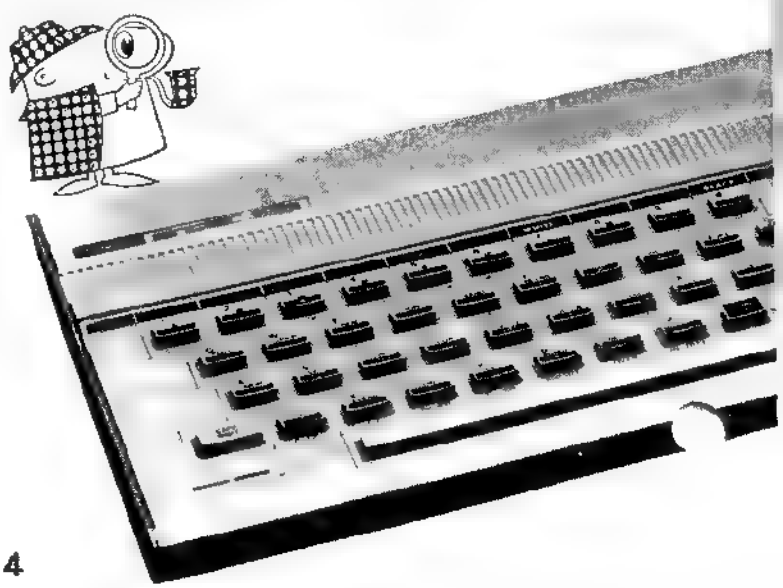
But we can't use GET_WORD, because we don't know whether it's in low or high memory! If we did, we wouldn't need to read VIDMOD in the first place! Fortunately, the SP register (the Stack Pointer) can get us out of this mess. This is because the stack always follows the RAM Res Code around in memory, so if the stack is in high memory, so is the code.

Unfortunately, there's no instruction that moves SP to another register. To get around this, LD HL,0000 and then ADD HL, SP. This effectively puts SP into HL, and let's us find where the RAM Resident Code is. The method suggested by Timex (using VIDMOD), can be very unreliable; you might consider using this method, instead.

As Strong As It's Weakest Link

If you've used an assembler on the TS2068 to write a machine code program larger than 2k or so, you've probably noticed that you have problems getting your source code to fit in the available memory. That's because a line of source text, which could take 10 or 20 bytes, will assemble into an instruction only 1 to 4 bytes long. While certain Spectrum assemblers have clever ways to get around this problem, we usually just break the code into smaller pieces, assemble them separately (usually with some modifications), and then link the separate pieces together by hand.

It should be no surprise that the assemblers used on some of the computers that the "big kids" use, can do this linking automatically. Code is assembled in separate "modules", with special reference commands for labels that are actually pointing to an external module. These separate modules are then linked by a program that's unimaginatively called, a "linker". Those of you who've seen the Timex listing of the "TS2000" ROM code will have seen how this works. (Since it makes the code harder to follow, you've probably cursed it, as well.) Still, this allows a computer to assemble a program that's even as large as it's full memory capacity.



```
4200-62AD Function Dispatcher - Performs a CALL or JP from any bank to some
F9C0-FA6D   HOME & EIROM bank routines
 SAMPLE USAGE - LD   DE,00000
                PUSH DE             ;Set input & output params
                PUSH DE             ;to zero
                LD   DE,function #  ;From Tech Manual Table 3.3.4-2
                PUSH DE             ;MSB=1 for JP, 0 for CALL
                     (here we set up the registers & such as
                      if CALLing the routine from HOME bank)
                CALL 0x6200 or 0xF9C0
NOTES:Routine address lookup table only points to low-memory addresses for
 RAM-Resident Code routines. DO NOT try to use the Function Dispatcher to
 access another RAM Resident routine if they are located in high memory !!!
 ---------- ---------- ---------- ---------- ---------- ----------
62AE-6306 Link to Interrupt Handler in HOME ROM. Allows keyboard interrupt
FA6E-FAC6   routine at HOME ROM 02E. to be accessed, if HOME ROM does not
            control chunk 0, but chunk 0 memory links to this
NOTES:Can be modified to point to a different routine, but beware of 5 pairs
 of bytes that get changed when this routine is relocated
 ---------- ---------- ---------- ---------- ---------- ----------
6307-6314 Copy of NMI handler in HOME ROM, Not used at all
FAC7-FAD4
 ---------- ---------- ---------- ---------- ---------- ----------
6315 BS_MAX_BNK  A copy of the MAXBNK system variable. Allows access to this
FAD5             parameter, even when HOME Bank does not control Chunk 2
 ---------- ---------- ---------- ---------- ---------- ----------
6316-633A GET_WORD Performs an effective LD HL,(HL) where (HL) is an address
FAD6-FAFA   in any bank; not necessarily one that is active
 SAMPLE USAGE - LD   HL,address
                LD   B,bank #
                CALL 0x6316 or 0xFAD6
 ---------- ---------- ---------- ---------- ---------- ----------
633B-6358 PUT_WORD Performs an effective LD (HL),DE where (HL) is an address
FAFB-FB18   in any bank; not necessarily one that is active.
 SAMPLE USAGE - LD   DE,word to be sent
                LD   HL,address
                LD   B,bank#
                CALL 0x633B or 0xFAFB
NOTES:This routine contains BUGS. TM6.5.3 gives adequate corrections. These
 corrections will relocate properly.
 ---------- ---------- ---------- ---------- ---------- ----------
635C-63AC WRITE_BS_REG Writes the value in register E to Bank Switching
FB1C-FB6C   Register whose number is in D. Avoid using this routine. Let
            the RAM Resident Code access it for you.
NOTES:For those who'd like to change this to drive a newer hardware archi-
 tecture, the original code is located at X1ISC-X11AC & copied to HOME RAM
 ---------- ---------- ---------- ---------- ---------- ----------
63AD-6404 READ_BS_REG Reads Bank Switching Register (single nybble) whose
FB6D-FBC4   number is in register D, & another from register whose number
            is in register E. Packs both into E register. Avoid using this
            routine. Let the RAM Resident Code access it for you.
NOTES:For those who'd like to change this to drive a newer hardware archi-
 tecture, the original code is located at X11AD-X1204 & copied to HOME RAM
 ---------- ---------- ---------- ---------- ---------- ----------
6405-644C GET_STATUS Gets the Horizontal Select Byte (lo-active) for the
FBC5-FC0C   desired bank into the C register. If it's an expansion bank,
            the status will also be returned in the B register
 SAMPLE USAGE - LD   B,bank #
                CALL 0x6405 or 0xFBC5
NOTES:Because of the flakey way that I/O port F4 is used as a Horizontal
 select for all 3 standard banks, they may "claim posession" of chunks
 actually controlled by expansion banks. This routine should be used by a
 larger one, that checks all expansion banks as well as standard banks, and
 uses the information as a coherent whole. Also, this routine contains BUGS
 corrected in TM6.5.2, though they can't correct the above problem.
 ---------- ---------- ---------- ---------- ---------- ----------
644D-645D GET_CHUNK Computes the chunk for a given address
FC0D-FC1D
 SAMPLE USAGE - LD   HL,address
                CALL 0x644D or 0xFC0D
                  (return with A=Horiz Select Mask - hi-true)
 ---------- ---------- ---------- ---------- ---------- ----------
645E-649B GET_NUMBER Returns the bank # for a given address
FC1E-FC5B
 SAMPLE USAGE - LD   HL,address
                CALL 0x645E or 0xFC1E
                  (return with A=bank #)
NOTES:Handles "oddness" in GET_STATUS by checking expansion banks first. Has
 a BUG, per TM6.5.4. Should not be used if there's a chance the EIROM might
 be in use.
 ---------- ---------- ---------- ---------- ---------- ----------
6499-651D BANK_ENABLE Gives control of desired chunks to specified banks
FC59-FC9D
 SAMPLE USAGE - LD   B,bank #
                LD   C,horiz select(lo-true)
                CALL 0x6499 or 0xFC59
NOTES:Contains BUGS per TM6.5.4. The manual gives an adequate fix, but a
 neater fix would be to put F3 at 649A, and FB at 651B. Also, errors in the
 EIROM relocation table PREVENT THIS ROUTINE FROM RELOCATING PROPERLY!
 ---------- ---------- ---------- ---------- ---------- ----------
651E-6549 SAVE_STATUS Used internally to save bank information before making
FC9E-FD09   temporary Horiz Select changes; eg, CALLing a routine in another
            bank.
 ---------- ---------- ---------- ---------- ---------- ----------
654A-6571 RESTORE_STATUS Used internally to put all banks back as they were
FD0A-FD31   before SAVE_STATUS was CALLed
NOTES:This will undo any video mode changes made since CALLing SAVE_STATUS,
 as well as other port FF control bits, per TM6.5.5. Contains a BUG, which
 can be fixed per TM6.5.4
 ---------- ---------- ---------- ---------- ---------- ----------
6572-658D GOTO_BANK Performs an effective JP to any bank. Does not pass
FD32-FD4D   any parameters.
 SAMPLE USAGE - PUSH address
                PUSH Bank#/Horiz select(lo-true)
                CALL 0x6572 or 0xFD32
NOTES:This routine acts like a JP, even though it's accessed through a CALL
 ---------- ---------- ---------- ---------- ---------- ----------
658E-65CD Bank Switching Stack - An additional stack, simulated in software.
FD4E-FD8D   Each time CALL_BANK is run, the return address and PAR_IN go here
 ---------- ---------- ---------- ---------- ---------- ----------
65CE-65CF BS_SP - The Bank Switching Stack Pointer. Used to simulate the Bank
FD8E-FD8F   Switching Stack
 ---------- ---------- ---------- ---------- ---------- ----------
65D0-660B CALL_BANK Performs an effective CALL to any bank, and contains pro-
FD90-FDAB   visions to pass parameters.
 SAMPLE USAGE - PUSH the parameters (# of bytes is called PARAM_OUT)
                PUSH address
                PUSH bank #/horiz select(lo-true)
                PUSH PARAM_OUT
                PUSH PARAM_IN
                CALL 0x65D0 or 0xFD90
NOTES:Contains a BUG which may be fixed per TM6.5.4. Also, PARAM_IN and
 PARAM_OUT represent the number of bytes; not the number of PUSHes
 ---------- ---------- ---------- ---------- ---------- ----------
660C-66E7 MOVE_BYTES Used only as a subroutine to XFER_BYTES, and is intended
FDAC-FDA7   to transfer bytes between banks when source and dest nation
            chunks overlap and the transfer is between two different banks.
NOTES:Contains numerous BUGS which are not documented in the Technical Manual.
 Due to the programmer's misunderstanding of the subtleties of LDIR and LDDR,
 and the differences in their usage, some counters are not properly updated,
 and some intermediate transfers can be made to the wrong part of the stack,
 destroying critical information. Major bugs can be tolerated by putting hex
 73 at 66DF, and hex 72 at 66E2. Still, this routine can only be used in the
 LDIR mode; fortunately, the LDDR case is not needed where this is used.
 ---------- ---------- ---------- ---------- ---------- ----------
66E8-6721 CREATE_BITMAP used only as a subroutine to XFER BYTES, and is
FDA8-FDE1   intended to produce a low-true "Horizontal Select" byte for all
            the chunks involved in either the source or destination bank for
            a data transfer.
NOTES:Contains undocumented BUGS. Due to improper computation of first and/or
 last bytes in a data transfer, this may give an improper result, when the
 error in computation straddles a chunk boundary. These can be corrected by
 inserting at 66F3 and following: hex 08,3C,20
 ---------- ---------- ---------- ---------- ---------- ----------
6722-6814 XFER BYTES An intelligent transfer routine to move data between
FDE2-FFD4   banks, but also intended to allow transfer within a single bank,
            wether or not all the necessary chunks are enabled.
 SAMPLE USAGE - PUSH source bank #/dest bank #
                PUSH source addr
                PUSH dest addr
                PUSH # bytes
                PUSH direction ;0000 - like LDIR, FFFF - like LDDR
                CALL 0x6722 or 0xFDE2
NOTES:Contrary to its description in the Technical Manual, this routine was
 intended to be able to do transfers between larger memory areas than just a
 single source chunk and a single destination chunk. The mentioned limitation
 was probably intended to mask one of the problems in this routine. Also, this
 routine does not relocate properly to high memory. This routine requires that
 the machine stack be in its proper location in the RAM Resident Code. It was
 not intended to be able to transfer data into or out of the chunk that
 currently contains the RAM Resident Code. If the stack is nearly full, the
 transfer will be aborted, without notifying the CALLing routine. A status
 flag, intended to perform such a warning gets corrupted before completion.
 The Technical Manual documents only one bug, but several changes are needed to
 get this working properly. Location 6722 gets 00, 6728 gets 08, 676A gets 3F
 (that one is the Tech Manual fix), 67C2 gets 2C, and 67FF gets 00. Because
 this code would normally be used to initialize expansion banks, it's possible
 that this routine would have to be fixed before they could be debugged. With
 these fixes, AF is no longer preserved, and A is now returned with a status
 code. It will contain 00 if the transfer was sucessful, and 01 if it was
 aborted, due to insufficient stack space. Because of the aforementioned and
 not readily correctable bug in MOVE BYTES, using this routine in the LDDR
 mode can crash the system in some cases. The only time the LDDR mode may
 really be needed would be certain times when the source and destination areas
 overlap WITHIN THE SAME BANK. This case does not cause the problem, so if the
 use of the LDDR mode is limited to this case, there will be no trouble.
 ---------- ---------- ---------- ---------- ---------- ----------
6815-6823 GOTO_EXT A routine intended for use only during initialization. Does
FFD5-FFE3   an effective JP (HL) to the EIROM. Would not work properly if
            expansion banks were enabled in chunk 0, hence its suitability
            only for system initialization.
```

Unfortunately, the addition of bank switching caused a problem that most linkers can't handle; there are two blocks of code with identical memory addresses. This means that the Timex folks had to go back to the old method of assembling (and linking) the Home ROM code and the EXROM code in two separate batches, and then linking them by hand. For a program of that size, this is an incredible problem. Every time the code is re-assembled, the hand linking must be done over again! And a program this size would get reassembled a lot. This just begs for a few spots to get "missed", and they certainly did.

We might expect to find these incorrect links where an instruction in one ROM references an address in the other ROM. We'd also expect that the incorrect address will be nearly correct, since it was probably correctly linked once, but the addition or deletion of a few instructions somewhere will have shifted everything in memory slightly. This is, in fact, only one way that mis-linking can make our lives miserable.

The earlier mention that some of the RAM Resident Code does not relocate properly to high memory is another example. You see, the EXROM contains a "relocation table", which is supposed to point to the various spots of the RAM Res Code that need changing. For example, the second and third bytes of a CALL instruction contain a memory address that must be changed if the code being CALLed gets moved.

Sadly, the programmers could have used labels in their assembly code to make the assembler produce a perfect relocation table. If we look at the end of the RAM Res Code listing (they call it the fixup table) in appendix A of the TS2068 Technical Manual, we can see that they instead chose to figure the numbers out by hand, and insert them directly into the code. Too bad.

Another EXROM table that didn't get fixed properly is the address table that the Function Dispatcher uses to find various ROM routines. Some (not all) of the dispatcher codes marked "reserved" in the Technical Manual actually point to a routine, but are off by a few bytes. Note that the majority of dispatcher codes reference the Home ROM. We'd expect that they'd use labels in the Home ROM assembly to generate most of the table, and then hand-patch in the addresses for the EXROM and RAM Res COde. Sure enough, the portion that points to the Home ROM is 100% correct, but the other two portions are a disaster.

To be fair, the Timex programmers tried to set up the EXROM in such a way as to reduce the number of mis-linking errors. The various tables and blocks of code in the EXROM are spread around, giving each one room to grow without encroaching on the space allocated by the others. Each block begins (or ends) at a nice, even hexadecimal number, and the space after (or before) each block is filled with FFs or 00s.

I've had several readers look at these gaps and give the fascinating suggestion that there may have originally been code there, which was blanked out prior to production of the ROMs. Since each gap bounds itself on a nice even hexadeciamal number, however, I must (sadly!) confess my doubts. Since each person who mentioned it also used DECIMAL, not hexadecimal addresses in their letters, I can see how this subtle, but important clue might have been missed. (C'mon guys! I said in Part 1 that we really need to work in hexadecimal here. You gotta trust me after all we've been through!)

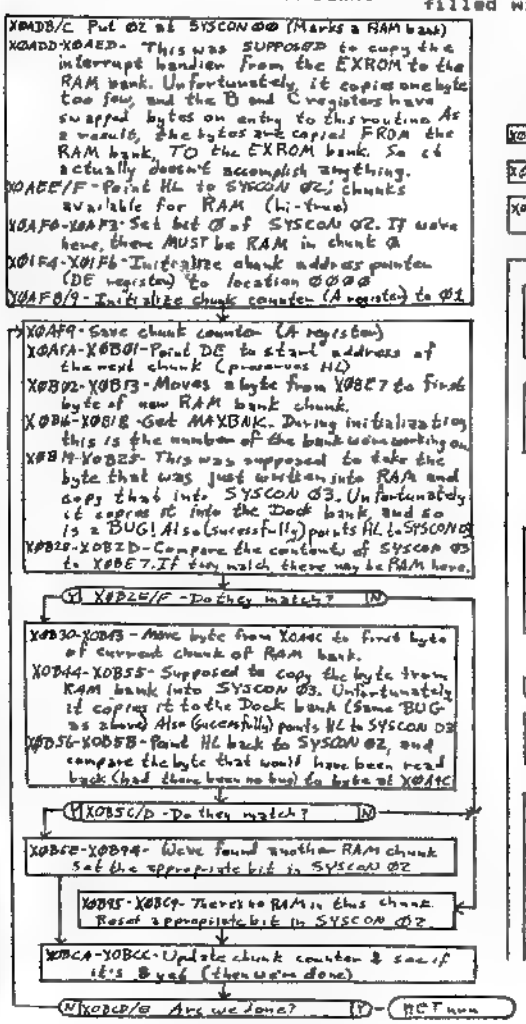
In any case, though we can't cover the fixing of the EXROM in detail, the following map should aid those who want to fix the tables, and make permanent changes to the bugs in the RAM

Resident Code.

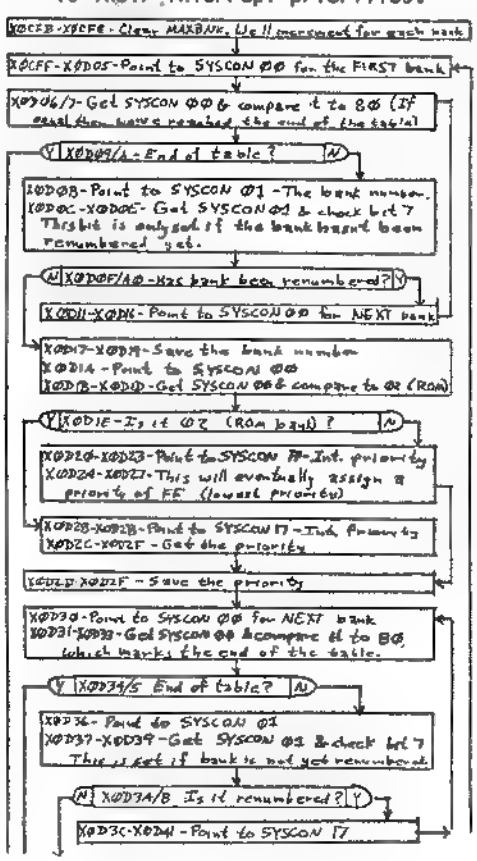
EXROM Map

0000-0067	- Misc. Housekeeping	
0068-08E6	- Cassette I/O	
08E7-0DAF	- Extensions to system initialization (We've flowcharted most of this)	
0DB0 0F42	Video Mode Change Routines	
0F43-0F89	- Passes list in extra BASIC commands FORMAT, CAT, MOVE, ERASE, LOAD *, SAVE *, onto the stack. Routine at 25B9 in Home ROM tries to CALL this, but is off by a few bytes. Likewise, this routine tries to CALL a Home ROM routine, with the wrong address.	Home ROM Overflow Block 0000-0FFF
0F8A-0FA7	- Performs crude interbank JPs and CALLs. Should be used only by the initialization code.	
0FA8-0FFF	- The block is filled out with zeros.	
1000 1623	- Initial RAM Resident Code is copied from here. A short stretch of FFs from 138E-13CF is the initial bank switching stack.	Initial RAM Res Code
1624-17FF	- The block is filled out with zeros.	
1800-1BFF	- Not Used (filled with FFs)	
1C00-1CFF	- Not Used (filled with zeros)	
1D00-( )	- Fixup table for relocating RAM Res Code. Address values start at 1D00. List grows UPWARD in memory.	
1D7A-1EDB	- Unused space between tables. (Filled with zeros)	
( )-1FFF	- Address table for Function Dispatcher. Starts at 1FFF and grows DOWNWARD in memory. This table is also broken into 3 sections; 1 for EXROM services, 1 for Home ROM services, and 1 for RAM Res Code services (low memory addresses only!!!) Unused space between these subsections is filled with FFs.	EXROM Tables

Flowchart 7:
Initialize a RAM bank.

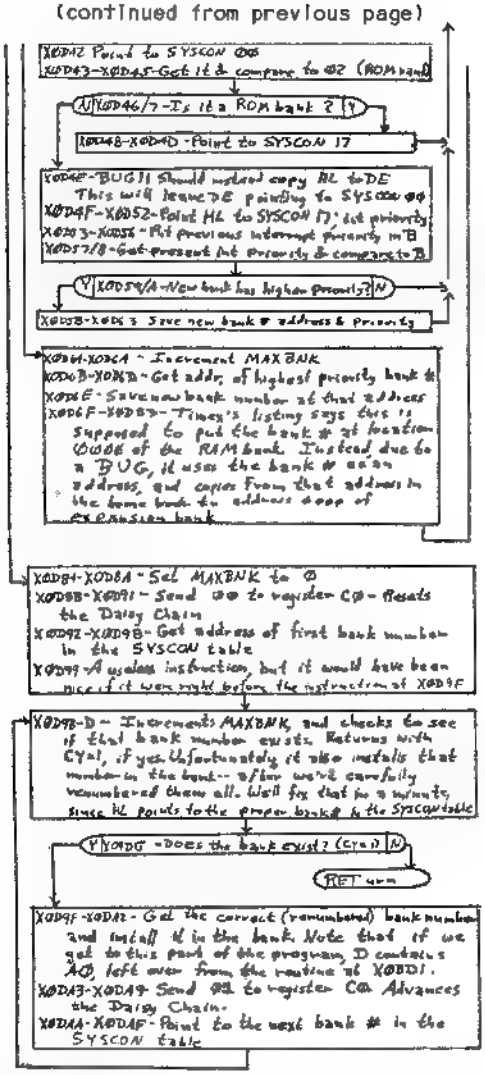


Flowchart 8:
Renumber banks according to their interrupt priorities.



Flowchart 8 continued...

Flowchart 8
(continued from previous page)



A Chip Off The Old Block

As we said last time, there are many portions of code in the Home ROM that are blocked off, so they're never executed. If we wished to use the "Sinclair Interface One" method of adding a disk, microdrive, or other I/O devices, we'd leave these blocked off. However, this would require extra hardware to switch in a "superbank", at the right moment. Since the blocked off code has the ability to link to the normal expansion banks without such extra hardware, it may seem attractive to try to restore that old code to working order. The following table gives a good feel for what they are, and what they do. In each case, the routines are blocked by a JR, JP, or RET instruction.

14BB - Would have allowed the execution of expansion bank code to OPEN a stream to a specially made channel.

2460 - Would have CLOSEd all 16 streams & rebuilt the SYSCON table after the execution of the BASIC command "RESET #". The rebuilt table would have been a "cold reset", using the EXROM routine at X09F4 (shown a long time ago in flowchart 4.)

2487 - Would have performed a "warm reset" of the SYSCON table after the BASIC command "RESET". This would have used the EXROM routine at X0C4C (not flowcharted).

24C0 - Would have run expansion bank code upon execution of the BASIC command "RESET # stream"

24EE - Would have passed the information in an extended LOAD or SAVE command (ie, LOAD # "D",list of information") onto the stack, and then CALLed a LOAD or SAVE routine, perhaps for a disk or microdrive, in an expansion bank. This would have CALLed the routine at 25B9, which is also never used. This is fortunate, because it tries to CALL an EXROM routine with a mis-linked address, and also has a RET command missing, following a CALL at 25DE

25E4 - This is part of the code that would have passed parameters from the BASIC commands CAT, FORMAT, MOVE, and ERASE. The blocking JP instruction, at 25E1, is where Timex deleted code to make room, as mentioned in the previous installment.

"Let Us Reconstruct Watson" . . .

An interesting item has been recently published in the Jan-Feb issue of the newsletter of the Long Island Sinclair Timex User Group. From its language and format, I suspect it's an early version of Timex's functional specification for bank switching! (They may have titled it differently, but that's the type of document it is.) Timex would certainly have had to make such a spec available to third party software developers. So, in hindsight, it's reasonable that a copy should eventually come to the surface.

Still, the person who "leaked" the document could come to some trouble for doing so. This may be the reason it was submitted under the pseudonym of "Dr. Watson". (I love it!!!) Well, whoever you are, Doctor, thanks a bunch. You've done a great service to the cause.

Because it's an early version, there are portions that have been superceeded by engineering changes in the machine. (See Part 3 of this series, where we discuss the bank switching tutorial in SAMS "TS2068 Intermediate/Advanced Guide".) As such, its description of the bank switching registers is not quite accurate. However, we do get a complete picture of the SYSCON layout, an idea of some of the peripherals Timex at least considered producing, and a description of the BEU.

We also see how some additional tables of data might have been written in home RAM by expansion banks during the power-on initialization. (For Spectrum users: these seem somewhat analogous to the extra information in the "m", "n", "t", and "d" channels used by the Microdrives, network, and RS-232 ports on the Interface One.) Since these tables are a function of the ex-

pansion bank software that doesn't exist yet, we can still redesign these as we wish. But it's still interesting to see what Timex had in mind.

The BEU was intended to allow an additional 4 devices to be plugged into the TS2068 (you're only supposed to plug two or less directly into the TS2068, according to the Technical Manual. If you needed more devices, you could plug ANOTHER BEAU into the first, to allow a total of 7 devices to be plugged in. A device could have been a peripheral, or could have contained one or more expansion banks. A device containing peripheral hardware AND an expansion bank to control it is referred to as an "intelligent device", by the way.

The list of device specifications (the ASCII characters that define a channel type) included the standard ones as shown in the Technical Manual, but also included specifiers for a Telecommunications device, a stringy floppy (like a Microdrive?), both floppy and hard disks, RS-232 and Centronics interfaces, an 80 column printer (it's uncertain how this would differ from the Centronics interface) and a network. Also the letter "M" is marked as reserved. This is what Sinclair uses for its Microdrive channels, but this is a stringy floppy device, so the purpose of this is uncertain.

The biggest bonanza from this document, however is the complete SYSCON table layout. As mentioned last time, the layout I gave was incomplete, and I began to give some corrections last time. A complete layout would not have been possible, because the ROM routines don't use all the SYSCON locations, and there's a conflict in various code locations as to how a certain set of initialization code is pointed to by the SYSCON table. As it turns out, I chose one possibility and Timex intended the other (curses!).

Furthermore, I speculated that certain expansion banks might contain complete replacements for the system ROMs. This means that certain "reserved" memory locations would contain JP instructions, so that instructions like RST 8 and RST 10 would work under these ROMs. Timex seemed to have no intention of doing this, but it's still possible, as far as I can see. In any case, here's the Timex SYSCON layout for an expansion bank, along with my original comments:

```
00 01=ROM  02=RAM  00=Inactive
01 Bank #. MSB is set if not yet renumbered

   The following is copied from 0000-0015 of ROM expansion banks
02 For RAM - Chunks available - High true
   For ROM - Channel specifier, if this bank controls a channel.
             This will be an ASCII character, and the initial-
             ization software resets bit 5, insuring that the
             letter will be uppercase.
03/4 Address of OPEN routine for the channel.
  (Alternately, 02-04 could have a residual JP instruction, which
  does no good to the SYSCON table, but allows RST 0 to work in
  the expansion bank, since the JP is also at location 0000 of
  that bank.)
05/06 Address of CLOSE routine, if the bank controls a channel.
             Call with RAM Res Code with PRM_OUT=2, and stream
             number on the stack.
07/08 Timex called this the address of the SELECT routine. It
             could have been used in initialization, and to
             attatch the current channel to this bank (?)
09/0A An I/O device INPUT routine address
0B/0C An I/O device OUTPUT routine address.
  (Alternately, 0A-0C could have contained a residual JP that
   would have been intended to allow RST 08 to work in the
   ROM bank)
0D/0E Address of Disk Command Handler routine
0F/10 Addr of device interrupt handler (72 bytes)
11/12 Addr of device initialization code  (Cold start)
13/14 Addr of device reset routine (warm start)
  (Alternately, 12-14 could have contained a residual JP that
   would have been used in the ROM bank to make RST 10 work.)
15 - Device type --  Bit 0 = 0 if bootable
                          = 1 if initializable
                     Bit 1 = 0 if non storage device
                             1 if storage (disk commands)
16 - Boot up priority. Low # = high priority. Home bank=80
17 - Interrupt Priority. RAM banks get 255. ROM gets lower
             value, which means higher priority

NOTE: The Timex document gives this list as ROM addresses, rather
        than SYSCON entries, as given here. The SYSCON
        displacements must always differ by two from the
        ROM addresses. This difference is not an error.
```

These are the items of real importance to those who'd want to implement extended bank switching on the TS2068. Other items in the document make interesting reading, however, and you may want to contact the L.I.S.T. group to see if back issues are available.

Final Thoughts

Throughout this series, I've made comments about the various bugs and defficiencies we've uncovered in the TS2068. While there is no denying that it does have numerous problems, we should see them in the perspective of the problems that likely faced the designers. Let's also not forget that the initial release of a computer will uncover many new bugs as a huge group of users tries things its designers hadn't condidered. (Wes' Second Law: It's unwise to buy Version 1.0 of ANYTHING.)

Remember that the TS2068 is a radical redesign of the Sinclair Spectrum with many new functions wedged in that its original designers never intended for it. It almost had to be forced to be able to do some of them. It was developed during a time when the home computer market was declining, and it ran way over schedule. It's engineers would have spent late nights in the lab, while getting called on the carpet during the day.

The mistakes we've seen are typical of the kinds I have observed (and even made!) in my many years as an engineer. Usually, they get fixed, but sometimes, there just isn't time. There's no reason to expect that those responsible for these bugs were not simply good engineers given a huge job on an impossible schedule. We can be thankful that they've accomplished what they have.

Writing the last article in a series always brings about mixed feelings. It's nice to see a job finished, but it's also like losing and old friend. I hope you'll continue to let me know about your own TS2068 projects...particularly any that deal with bank switching. I've made a lot of friends here, and I wouldn't want to lose contact.

I've been reluctant to do more than hint about the bank switching hardware that I've been playing with, all this time. The reason is that I wasn't satisfied with it, and was sure that many of you could provide a better way, given the proper food for thought. (And every now and then, I'm right; you've done spectacularly.)

For the record, I've modified the two low level communication routines in the RAM Res Code so that they instead communicate their data to a separate (and very small) Z80 computer. The second computer simulates most of the registers, and controls the horizontal select bytes for the expansion banks, which are otherwise controlled by the TS2068. This works, but it is a bit more complex than I'd like. If you'll look back over this series, and see my scattered hints about the virtues of changing the two low level routines (READ_BS_REG and WR_BS_REG) you may see the method in my madness.

Well, thanks to many of your suggestions, it now looks possible to modify these two routines so that they do all the simulation and control functions in their own limited space, under complete control of the TS2068, negating the need for a separate processor. This is still in its early stages, but extended bank switching could become much simpler, in the coming months...we'll see.

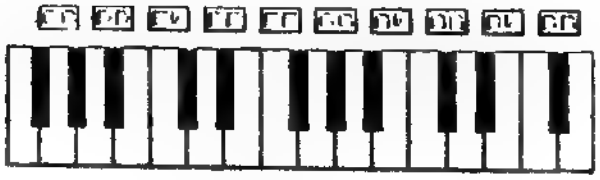
As I said, I don't want to lose contact. Please feel free to write to me: Wes Brzozowski, 337 Janice St., Endicott, NY 13760.

MUSIC DESIGN '2000'

FROM

AARON

SOFTWARE



HERE IT IS!!

A REAL MUSIC PROGRAM FOR THE 2068!

You can adjust the volume, tempo and duration of the beat and melody. You can record as you play and play it back later. You can string several tunes together and play them back continuously. You can save the program with the tunes you have written or entered. Room for over 1000 notes and pauses. Over 2 1/2 octaves! 10 chord keys!
If you want, you can use your own music books to enter songs of your choice.

Almost like having your own sythesizer on your 2068!
Nothing better for the price!

ONLY $19.95+$2 SH.

EXCLUSIVELY FROM:
RMG ENTERPRISES
1419 1/2 7TH STREET
OREGON CITY, OR 97045
503/655-7484

DO YOU HAVE PROGAMS ON CASSETTE THAT YOU WOULD LIKE TO HAVE ON YOUR AERCO FD-68 DISK?
WHAT WOULD YOU PAY FOR A PROGRAM THAT WOULD MOVE THEM FROM CASSETTE TO DISK AUTOMATICALLY?

Well we are here to announce just such a utility!
GET READY! HERE COMES——

S&K EXPRESS

Another GREAT utility from the author of such titles as:
KRUNCHER, KRUNCHER 1000 AND TRACER
This time we bring you EXPRESS, as in FAST!
We guarantee that this utility will save you a lot of time transfering programs to disk.

So simple to use! All you do is boot up the program and press PLAY on your tape recorder and watch the program LOAD and then automatically MOVE to the AERCO disk!
We couldn't make it any easier! AND THAT'S NOT ALL!
This program uses ABSOLUTELY NO MEMORY!
Once the transfer has been made, all that is needed is for you to reboot them, change the SAVE and LOAD commands and re-SAVE them in an auto-run mode.

The program includes a very good header reader and catalog function. It is just the first of many programs to come that utilize the bank-switching facilities of the 2068.

A GREAT ADDITION TO YOUR FD-68 UTILITY LIBRARY!

THE PRICE?
ONLY $14.95+$2 SH.

EXCLUSIVELY FROM:
RMG ENTERPRISES
1419 1/2 7TH STREET
OREGON CITY, OR 97045
503/655-7484

TRS WORKUP Co.
Delivers!

Let TOURIST C what HOT Z cannot.

Bank#255 Grp:SYNTWO Rtn:PASSEM

```
9657    01FEFE    LD BC,NN        65278 !Bank#254/Mask=01111111
9660    CD9964    CALL NN         25753 !CHANK_CHOICE..EXROM
9663    CD090F    CALL NN          3849 !CPASSAM (Really @ 3987)
9666    0100FF    LD BC,NN        65280 !Bank#255/Mask=00000000
9669    CD9964    CALL NN         25753 !CHANK_CHOICE..HOME
! This is an example of one of the barricades preventing the
!passing of parameters from BASIC statements.
```

Bank#255 Grp:SYNTWO Rtn:BADKEYS

```
9672    06CF      LD B,N            207: CAT
9674    180A      JR +12           9686
9676    06D0      LD B,N            208: FORMAT
9678    1806      JR +8            9686
9680    06D1      LD B,N            209: MOVE
9682    1802      JR +4            9686
9684    06D2      LD B,N            210: ERASE
9686    CD8928    CALL NN          10377
9689    2006      JR NZ,+8         9697
9691    CD6925    CALL NN          9577
9694    CD4418    CALL NN          6980
9697    C36725    JP NN            9575
```

Bank#255 Grp:SYNTWO Rtn:ERROR J

```
9575    CF        RST 8             ERROR
9576    12        DC                Invalid I/O device
! This barricade prevents using DOS, but ON ERROR can beat it.
```

TOURIST C
Delivers!

Tourist C presents a disassembly in a form you can USE! Decimal is shown when that is what the TS2068 requires for input. Hexadecimal is used in places where it is the better choice. Some redundancy is achieved. Both forms often show.

Relative branches show offset, code byte and destination.

Error calls give the exact error message and code byte.

Calculator calls show the routine in uninterpreted byte code.

WHAT MORE COULD YOU ASK?

You could ask for a data-base filing system. You got it!
You could ask for a universal printer interface. You got it!
You could ask for interpretation of code bytes. You GOT it!
You could ask for titling and a way to jump around. Guess what?
You also get "SPY"! Ah,well...

Did I mention..perhaps I..YES I really should. NO RELOCATION!

What you see above is an unretouched disassembly prepared by TOURIST C. The comments and emphasis of the titles were added by a word processer from the data-base file generated by TOURIST C. What good is a disassembler if you can't make notes? As you can see, getting down to the business of working with machine code is never just a matter of printing out all the raw data. You need better, and deserve it. TOURIST C gives it to you. The fact that TOURIST C bank-switches has nothing to do with skimping on the essentials. What it can show you in other banks is gravy. GET IT! TOURIST C Order#TS2SPY68B $32.50 + 1.50 S&H

Renew Your Subscription Today!

REVIEW

Spectrum

Turbo Esprit

Michael E. Carver

Durrell has done a decent job of providing an inexpensive means of driving one of the finest sports cars produced, the Lotus Turbo Esprit. To add an extra dimension to the simulation, Turbo Esprit is also a game of "cops and robbers". Actually, it is a game of "cop and drug smugglers". The plot of the game is to chase down four drug-runners and their armoured car supplier. The action takes place in one of four different cities. A city must be chosen upon loading, and to try a different city one must reload the game.

The display is similar to Durrell's Combat Lynx (a combat helicopter simulation). The screen is divided into two sections -- the dashboard and the playing field. Once again, the player's vehicle is seen on the screen from a view point just behind the car in a 3 dimensional world. The 3 D effect of the city landscape is effective, but a little jerky while driving at slower speeds. The car is, as one would expect, a right-hand drive ala England, and the city streets are laid out to be driven on the left-side of the road. This took quite some time to get used to, especially when making left or right hand turns. The city is made up of various 2 to 6 lane streets with a number of one-way streets. Some of the hazards encountered, while cruising the city, are pedestrians crossing in crosswalks, potholes (which are set off with barricades), and workmen on ladders, which extend into the street. Of course, what driving simulation would be complete without lane changers, who don't use their turn signals? Yes, the cars driving through town have operating turn signals.

Your car is equipped with a gun, which can be fired at other cars. Drug-running cars can be stopped by shooting them or continually rear-ending them. Points are scored for apprehending the drug cars, with a higher score provided when disabling the car by rear-ending it. There are penalties for ramming innocent cars or running over pedestrians. You are provided with four separate cars, as your car can be demolished by crashing into other cars or walls. It is also possible to lose a car when shot by one of the cruising hit-cars. One also scores points by taking out the hit-cars. There are also various gas stations which are needed when the fuel gauge gets low or your engine develops a misfire.

A map of the city is available (on the screen), marking the locations of gas stations and drug cars. There are various levels of play which effect the speed of the cars and the number of times they must be rammed before they submit. Control is via keyboard (which is user-definable) or joystick. By being able to define the keys, control by keyboard is easy and responsive.

As a simulation, the game has a good "feel", (especially when taking corners too fast). It is great fun to re-enact some of the great chase scenes from the movies (Bullet or the French Connection), by driving on the wrong side of the road to get around traffic jams, running red lights at busy intersections and dodging pedestrians. The plot and action of the game can get a little old-hat. It is not a game that I think one will get hooked on and forget to eat or sleep, but is great fun to pull out once in awhile and pass a few hours with. The action does not seem to slow down with the number of active objects on the screen, though it does suffer attribute bleeds. Also, one can occasionally get stuck, because some idiot driver went down a one-way street the wrong way.

Turbo Esprit is available from Curry Computer for $16.95.

PROGRAMMING CONCEPTS by Albert F. Rodriguez

The following article deals with programing on a Sinclair ZX81 (or TS1000). It is just a portion of a large document, with the remainder to be published in the next issue of TDM, along with a program listing. The listing is a game program, "ZX81 TIC-TAC-TOE", which will serve as the chief example and will be discussed extensively. If readers would like to get a "head start", a complete listing of the program, declarations and array content are available for $6.20 ppd.; or a cassette is a available (nonlistable) for those who don't want to key in the program, for $12.00, from the author. Albert F. Rodriguez, 1605 Pennsylvania Ave. #204, Miame Beach, FL 33139. (Foreign buyers add $2.00 for the cassette, or $1.00 for the listing).

(con't from last issue)

From this observation I deduced that it would be prudent to locate my subroutines, as often as it was practicable, below and as near to the place that they have to be called from in order to acheive an optimal MC/PS ratio. This technique can be best implemented, given a program that is relatively as multi-functional as is mine, by using what I call: "drivers". (For this term, but not the meaning given to it below, I am grateful to Mrrs. Frank L. Friedman/Elliot B Koffman, Problem Solving and Structured Programming In Fortran, Addison-Wesley Publishing Co., Inc., 2nd Edition, 1981, Page 299.)

A driver is similar to a "main program" (to learn about the concept of a main program see the chapter on Subroutines in the User Manual).
A driver calls one or more subroutines (or another driver(s)) within an overall program in that it is actually a suproutine that may called by a main program (e.g., see the driver routine Game in my program, which is called from line aa9 and consist of lines 7001-7011).

There should be only one main program within an overall program, but there can be many drivers within an overall program. The unique value of a driver is that it allows a subroutine or another driver, far below in the overall program, to be nearer to the actual place from where it is being called within the overall program. And this, indeed contributes to a better MC/PS ratio.

The proceedural rules derived from structuring (as effeciently as possible) the main program (see lines 112-120), drivers and subroutines (see Declarations below for exact line numbers) within my overall program are, for convenience sake, referred to as "Rules of Top-Down Design," and can be summarized as follows:

1. A main program is placed immediately after the program's name and any commands/initializations, if any, that appear at the beginning of the instruction area.
2. A main program begins with either a subroutine or a driver call, and it ends with a GOTO statement.
3. A main program calls either drivers or subroutines; it best calls itself with a GOTO statement.
4. A driver or subroutine are located best when they are below and nearest to where they are being called.
5. A driver or subroutine are located, in an overall program, in the order that it is first called by a main program or another driver.
6. If a driver and main program bothcall the same driver or subroutine , then, this same driver or subroutine is located in the overall program in the order first called by the main program.
7. A driver calls either one or more drivers or subroutines.
8. A driver or subroutine should contain at least one RETURN statement.
9. A driver or subroutine best calls itself by using a GOTO statement.
10. A subroutine, to be considered as a subroutine, must not contain either a driver or another subroutine.
11. Both a driver and subroutine are best called b a GOSUB statement unless they each call itself.
12. Always use the smallest line number possible when writing each line of a program.

So far I have been elucidating some of the proceedural techniques used in writing and structing my program. This presentation, however, would be incomplete without an explanation of what each particular section (form top to bottom) of the overall program does and why it was written. It is toward this end that I dedicate the rest of this work.

To clearly know what is being discussed next, the reader should have nearby a copy of the program list and its declarations (see below). The actual data stored within certain arrays in the variable store is not necessary to understand what follows. (A complete listing of the program, declarations and array content are available for $6.20, ppd, to whomever may decide to key in this program themselves rather than purchasing it in cassette form for $12.00. Foreign buyers add $2.00 more for the cassette and a $1.00 more for the listing and materials.)

The programm, on tape, is not listable on the screen. This precaution was taken so that a user would not accidentally disrupt the program if he/she happened to gain access to the code area of the program. To avoid reloading the program a user, whenever he/she has access to this area, should key in GOTO 7 to restart the game (see Profile sheet for instructions about how to stop and restart this program). No-mony-back-guarantee are the terms applicable to whomever buys this progra in cassette form or not.

More of this article next issue.

ZX81 Data Acquisition Module

PART II

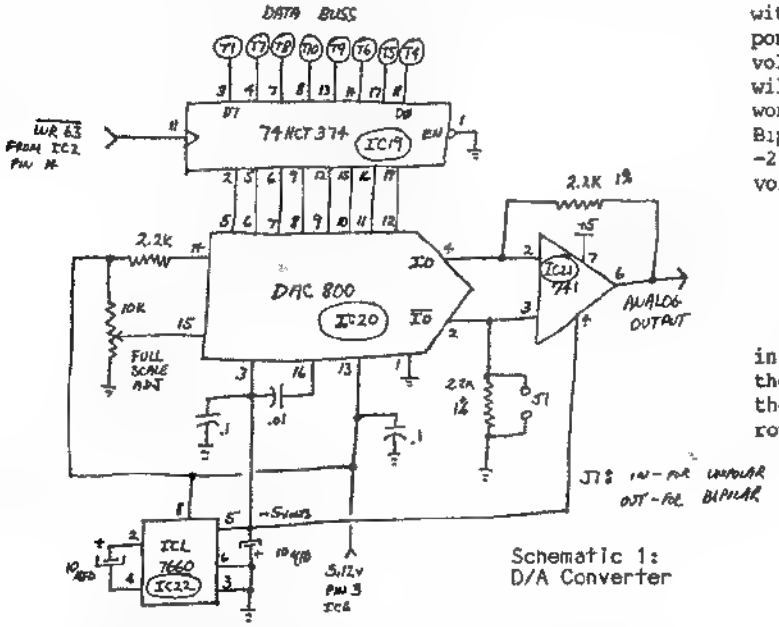
By

Tim Stoddard

As promised last issue, we'll be trying our luck at building a digital to analog converter (D/A for short). I also got to thinking (a sometimes dangerous activity). What's the use of all this fancy A/D and D/A stuff without some reference to time? For instance, in robotics, you would send an analog signal to a motor for a "specified time" to move, say, an arm or leg. You could use timing loops in the software, but that is inherently inaccurate. What is needed is a REAL-TIME CLOCK. So, in this issue we will be constructing a Real-Time Clock in addition to the D/A converter. Although the Real-Time Clock is on the same board as the D/A and A/D, I designed the clock with it's own selection logic, so if you want to use only the clock, it could be done very easily.

D/A CONVERTER

Schematic #1 diagrams the circuit. IC19 (74HCT374) was shown last issue in the A/D circuit as an optional port. We will be using this port to supply the D/A converter with it's 8-bit byte to be converted to an analog voltage. IC20 (DAC 800) is the workhorse that does the actual conversion. It comes in many variants (DAC 800/801/802). They differ in conversion errors and maximum temperature range. For our purpose, any of them



Schematic 1: D/A Converter

will do. In fact, my prototype uses the most inaccurate (DAC 801=full scale nonlinearity of .39%). I don't think I'll complain about .39%! IC21 (741) converts the "current" output of the DAC 800 to a voltage output and also acts as a buffer. The one curve ball that did come up while I was designing this was: the DAC 800 and it's variants use THREE supply voltages! I strongly dis-like anything other than "plus five volts", but in order to bridge the digital and analog world, we'll just have to grin and convert it. Anyway, the DAC 800 needs +4.5 to +18, -4.5 to -18, and a reference voltage for the current switches. IC22 (ICL 7660) converts the +5 volt supply to -5 volts for both the DAC 800 and the 741. I used this approach rather than an external power supply because, as it is, my bench is already cluttered with power modules. The one thing we don't need is another power module hanging out of the wall!

The DAC 800 works by taking the input byte, and then using each bit to switch a binary-weighted current source. For instance, bit 7 represents decimal 128, or half the value of the input byte. When this bit is a one, the current switch attached to that bit in the DAC 800 will supply enough current to generate half of the reference voltage via the 741 op amp.

Before wiring up the circuit, you'll need to decide if you want "unipolar" or "bipolar" (unipolar = 0v to +5v,; bipolar = -5v to +5v) output. Then ground pin 3 of the 741 and pin 2 of the DAC 800 directly for unipolar operation, or for bipolar operation, through a 2.2k resistor. If you do decide to go with bipolar operation, use 1% resistors for the two 2.2k's attached to the 741 op amp. These can be purchased at Radio Shack as part no.271-309 for $2.49 (actually a package of 50). This will improve symmetry. After wiring up the circuit, a single adjustment should be make. Output 255 decimal to the DAC 800 port; in our case this is port 63H (use the BASIC/MC program below). Next, adjust the full scale pot for the desired full scale output. NOTE that the current switches need about 1.5 volts to switch, so full scale output should be 1.5 volts LESS than the supply voltage to remain linear. I set mine up for 2.56 volts...this then gives .01 volts per count...an easy number to work with. For example, sending decimal 100 to the DAC 800 port will cause the DAC 800 to output via the 741, 1.00 volts, or if you send decimal 197 to the port, the 741 will output 1.97 volts. You can see how easy this is to work with. This of course, assumes unipolar operation. Bipolar operation would give an output voltage range of -2.56 to +2.56 volts, and increase each count to .02 volts.

The ML routine to write the port is very easy and follows:

```
16514   62      00      LD A,n  ;load A with data to output
16516   211     99      OUT 63,A ;output the data to DAC
16518   201             RET     ;return to basic
```

To use the routine simply POKE the above MC routine in a REM statement, then within the BASIC program, POKE the desired data byte to output into location 16515 and then execute the routine. Here is a sample of the above routine:

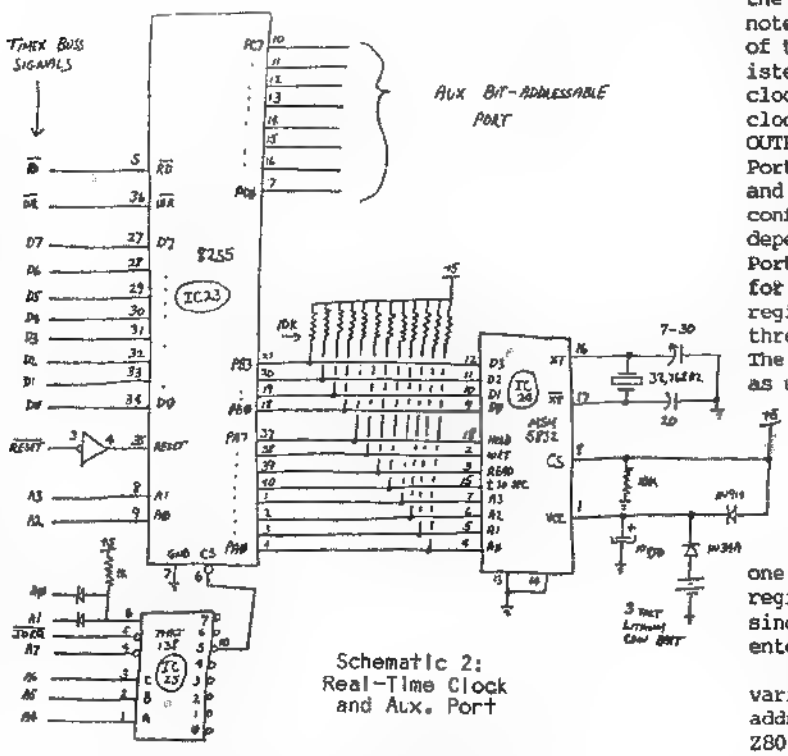
```
5 REM ***** <== Poke the above ML routine here
10 PRINT "ENTER OUTPUT DATA"
20 INPUT D
30 POKE 16515,D
40 RAND USER 16514
50 GOTO 10
```

REAL-TIME CLOCK

This part of the project is perhaps, the most useful. It can be constructed outside the DAM board project and used separately. Everything from games in "real-time", to timed control of BSR modules in the home can be accomplished using this clock.

The circuit is very simple and also features a bit-addressable port. IC23 is a Programmable Peripheral Interface (or PPI for short). It replaces three 8-bit

ports and selection logic, and is fully programmable. All three ports can be programmed as input or output and in the case of port C can be programmed as a bit-addressable output port. Fortunately, it is also CHEAP ($1.69 from JDR). IC24 is the actual clock chip and is a MSM5832 ($2.95 from JDR). The crystal is a 32.768KHZ unit and costs $.95 (also from JDR). IC25 does our I/O decoding and is the familiar 74HCT138. Note, you can also use the 74LS138, but it will consume more power. I HIGHLY recommend using the battery back-up, unless of course, you want to set the clock each time you "power-up". A nice lithium coin-type battery holder and battery is available from JDR (holder $1.95; battery $3.95). The trimmer cap on the clock is available from Radio Shack, as are most of the other passibe parts. The inverter used to invert the RESET signal off of the TS/ZX buss, is an unused section of the 74HCT04 in the A/D circuitry. The pull-up resistors are needed because the MSM5832 is a CMOS device where the 8255 is not.



Schematic 2:
Real-Time Clock
and Aux. Port

The software to use the clock is a little more complex than we have done up to now, due to the use of the 8255 PPI. However, it's nothing that can't be overcome. Generally what we need to do is configure the three ports, then send the appropriate data to each of the ports as needed. Looking at schematic #2, you'll note that we use two of the ports for the exclusive use of the clock. Port "A" is used to send the clock register address and four control signals to control the clock. This port then will be used to send data to the clock only. Therefore, we will configure this port as an OUTPUT port while reading or writing the clock chip. Port "B" on the other-hand, will be used to both send and receive data to the clock chip. Therefore, we will configure this port as either an INPUT or OUTPUT port depending on the operation being performed on the clock. Port "C" is not used by the clock and can be configured for your particular use. The 8255 has four possible registers that can be addressed; one for each of the three ports and one that is used to control the 8255. The following Table illustrates the register addresses as used in the DAM board.

8255 PORT#	8255 REGISTER ADD	Z80 HEX PORT#	Z80 DEC PORT#
A	0	53	83
B	1	57	87
C	2	5B	91
CNTRL	3	5F	95

There are actually two MC routines for this clock; one to set the clock and one to read each of the MSM5832 registers. All executable code is shown in decimal form since this is the most common way that MC routines are entered.

The following is the "jump table" for accessing the various routines to use the MSM5832. The table loads the address and READ control signal for the MSM5832 into the Z80 'H' register and also loads the digit MASK into the Z80 'L' register. The MASK is used to remove the unused bits from the particular MSM5832 register we are reading (each MSM5832 register is a 4 bit register).

```
16514   201 201>201  JP  $          ; Used in part 3
16517    24  91      JR  $5B        ; jump to set routine
16519    33  15  44  LD  $2C0F      ; set-up for YEAR10
16522    24  68      JR  $44        ; get YEAR10 dig.t
16524    33  15  43  LD  $2B0F      ; set-up YEAR1
16527    24  63      JR  $3F        ; get YEAR1 digit
16529    33   1  42  LD  $2A01      ; set-up for MONTH10
16532    24  56      JR  $3A        ; get MONTH10 digit
16534    33  15  41  LD  $290F      ; set-up for MONTH1
16537    24  53      JR  $35        ; get MONTH1 digit
16539    33   3  40  LD  $2803      ; set-up for DAY10
16542    24  48      JR  $30        ; get DAY10 digit
16544    33  15  39  LD  $270F      ; set-up for DAY1
16547    24  43      JR  $2B        ; get DAY1 digit
16549    33   7  38  LD  $2607      ; set-up for WEEK
16552    24  38      JR  $26        ; get WEEK digit
16554    33   3  37  LD  $2503      ; set-up for HOUR10
16557    24  33      JR  $21        ; get HOUR10 digit
16559    33  15  36  LD  $240F      ; set-up for HOUR1
16562    24  28      JR  $1C        ; get HOUR1 digit
16564    33   7  35  LD  $2307      ; set-up for MINUTE10
16567    24  23      JR  $17        ; get MINUTE10 digit
16569    33  15  34  LD  $220F      ; set-up for MINUTE1
16572    24  18      JR  $12        ; get MINUTE1 digit
16574    33   7  33  LD  $2107      ; set-up for SECONDS10
16577    24  13      JR  $0B        ; get SECONDS10
16579    33  15  32  LD  $200F      ; set-up for SECONDS1
16582    24   8      JR  $08        ; get SECONDS1 digit
16584    33  12  37  LD  $250C      ; set-up AM/PM/24 f
16587    24   3      JR  $03        ; get AM/PM/24 flags
16589    33   4  40  LD  $2804      ; set-up LEAP flag
;
; This routine leaves the clock digit in the BC register for
; use by the BASIC program.
```

```
16592     62 130     LD A,$82       ; set-up PA=out,PB=in
16594    211  95     OUT $5F,A      ; write control register
16596    124         LD A,H         ; write add and cntrl
16597    211  83     OUT $53,A      ; to clock via PA of 8255
16599    219  87     IN A,$57       ; read clock digit from
                                    ; PB of 8255
16601    165         AND L          ; mask off needed bits
   02     79         LD C,A         ; save clock digit in C
   03      6   0     LD B,$00       ; clear B
16605     62   0     LD A,$00       ; turn off add and cntrl
16607    211  83     OUT $53,A      ; to clock via PA of 8255
16609    201         RET            ; return to BASIC
;
;
;This routine will set the clock from a BASIC variable called
;D$. The variable MUST contain 11 digits as follows:
;
;           "YYMMDDWHHXM"
;            111111 1111
;            0 0 0  0 0
;
;As an example-----> LET D$="87020430854" will set the clock for
;WED FEB 4,1987 08.54:00
;
;Y10 (YEAR 10's)=8
;Y1  (YEAR 1's  =7
;M10 (MONTH 10's)=0
;M1  (MONTH 1's) =2
;D10 (DAY 10's) =0  NOTE add 4 to this digit for LEAP year
;D1  (DAY 1's)  =4
;W   (WEEK) =3 NOTE day of the week starting with SUN=0
;H10 (HOUR 10's)=0  NOTE: add 0 for AM,4 for PM, or 8 for 24HR
;                         if digit should be a "2" and it's a
;                         24HR clock make digit an "A"(2+8=A).
;H1  (HOUR 1's) =8;M10 (MINUTE 10's) =5
;M1  (MINUTE 1's)  =4
;NOTE that seconds are set to 00 when writing the clock chip.
;
;To set the clock for WED OCT 19, 1987 23.35:00 LEAP YEAR use
;
; LET D$="8710593A335"<NL>
; RAND USR 16517<NL>
;
; The routine will generate two different errors;

      ERR 2            The normal variable undefined error
      ERR E (inverted) This indicates that the D$ variable is
            E          NOT 11 digits in length.
```



```
16610     33  20  65  LD HL,$4114    ; load HL with add of
                                     ; variable to search for
16613     34  22  64  LD ($4016),HL  ; also put it in CH_ADD
16616    205  28  17  CALL $111C     ; find variable routine
16619    218  75  13  JP C,$0D4B     ; if variable not found
                                     ; ERR2
16622     35          INC HL         ; point HL to var length
16623     62  11      LD A,11        ; compare length to 11
16625    190          CP (HL)
16626     40   2      JR Z,2         ; jump ahead if=11
16628    207          RST 8          ; if not=11 generate an
16629    141          ADC A,L(INV. E); ERRE
16630     35          INC HL         ; point to
16631     35          INC HL         ; begining of string
16632      6  12      LD B,12        ; transfer count+1
16634     62 128     LD A,$80       ; program 8255 for
16636    211  95     OUT $5F,A      ; PA=out,PB=out
16638    120          LD B,A         ; transfer count is also
                                     ; used as clock address
16639    246 128     OR $80         ; add hold cntrl signal
16641    211  83     OUT $53,A      ; write port A
16643    126          LD A,(HL)      ; get digit from D$
16644    222  28     SBC A,28       ; string digit -28
16646    230  15     AND $0F        ; mask off bits 0-3
16648    211  87     OUT $57,A      ; write it to PB of 8255
16650    120          LD A,B         ; then strobe
16651    246 192     OR $C0         ; the write signal
16653    211  83     OUT $53,A      ; high
16655     35          INC HL         ; point to next D$ digit
16656     16 236     DJNZ,EC        ; loop for next
16658    201          RET            ; return to BASIC
16659      0          NOP            ; used in testing
16660     41          ADD HL,HL(D)   ; search table
16661     13          DEC C ($)      ; (D$)
16662    255          RST 38         ; termination byte
```

The following routine will allow you to enter and then use the MC routine as a BASIC clock. Note, however, that since BASIC is being used to access the clock, the update of seconds will be SLOW! You can, of course, include colons and whatever other "pretty-printing" you desire. I'll give a listing for a Machine Code clock in an upcoming issue of TDM (when another installment of the DAM board will be presented...mainly software).

Continued Next Page...

```
 1 REM *********      <<==== enter 170 *'s here
 5 FAST
 6 INPUT X
10 FOR X=X TO 16662
15 PRINT AT 20,0;X;" ";PEEK X;
20 INPUT D
30 POKE X,D
35 PRINT AT 20,10;PEEK X
36 SCROLL
40 NEXT X
50 CLS
52 SLOW
53 PRINT "ENTER DATE,TIME"
54 INPUT D$
55 RAND USR 16517
56 PRINT AT 10,10;
60 FOR X=16554 TO 16579 STEP 5
70 PRINT USR X;
80 NEXT X
90 GOTO 56
```

RUN the program, and when you get the "L" cursor, enter the starting address for the MC routine which, in our case, is 16514. After you enter all the bytes, you will get the "L" cursor. Enter the date and time in the format described in the MC routines, then the BASIC routine starting at 50 will display time.

That's it! As I've said in previous articles...Drop me a line! Let me know how you used the project. I'll be more than glad to assist anyone. We Sinclair hackers always seem to stick together. If you've always wanted some peripheral for your Sinclair but just can't seem to find one, let me know what it is. It may make a great future project/article in TDM. Don't forget, next issue will feature my INTERNAL 64k RAM upgrade for the TS1000. Watch for it!

TS 1000/1500 PROGRAM CHAINING

CONCLUSION

BY

Earl V. Dunnington

A few "bugs" crept into the manuscript and the text of Part Five as published in the last issue. The "tree" in line 4 of the second paragraph, on the left side of page 40, should read "three". The "TS1000" in the first line of the last paragraph, also on the left side of page 40, should be "TS1500". The "needed to save memory" starting on line 13, from the bottom of the right column on page 40, should read "needed and to save memory".

Having completed the homework assigned in Part Five, we are now ready to determine the Upper Limit of the Safe Area, the address of E_Line, the Lower Limit of the Safe Area, for each module and the minimum address to which we can lower RAMTOP and have the entire Chained Program operate properly. We will start with the TE module.

Turn on the computer and set RAMTOP to 17408, by entering the direct commands:

```
FAST
POKE 16389,68
NEW
```

Load the preliminary TE module from your homework tape using the direct command:

```
LOAD "TE"
```

When the cursor appears, type in one 32 character line (the left hand quote symbol marks the end of the line), then press ENTER. Type in STOP (using the shifted A key), then press ENTER. When the prompt appears, press any key. When the diagonal LOAD lines appear, press BREAK.

To determine the address of the Upper Limit of the Safe Area, starting at RAMTOP minus 50 (in this case 17408-50=17358), PEEK each address downwards in memory, noting the decimal value returned, until you find nothing but zeros for at least ten addresses. The command to be used is:

```
PRINT PEEK nnnnn
```

where the "n's" are the address to be PEEKed. The address just before those with all the zeros should be 17353 and should contain the decimal value 125. As the values used in the Machine Stack and the GOSUB stacks are pairs, in this case the High Byte occupying the higher uneven address (because RAMTOP is set to an even value), the Low Byte is zero in address 17352 and the Upper Limit (UL) of the Safe Area is the address 17351 for the TE module.

To determine the addrss of E_Line use the direct command:

```
PRINT PEEK 16404+256*PEEK 16405
```

The value returned should be 16918.

To determine the Lower Limit of the Safe Area and to overcome the curve that the Wicked Wizard of ROM has presented us with, we will use a slightly different "Flypaper" program than the one presented in Part Two of "Adventures In The RAM Jungle And Other Mysteries" (page 12, Nov/Dec '85 issue of TDM). Instead of POKEing a 5 into the addresses above E_Line, we will POKE a series of numbers from E_Line up in memory, using the Flypaper program (Figure No.5). After loading and operating the TE or other module, the top of the Calculator Stack can be found where the series is broken. The Lower Limit is the address just above this, where the value has not been changed.

If your computer is still on, use NEW to clear the memory, other wise turn on the compute and reset RAMTOP to 17408 as you did before. Type in the program of Figure No.5. Enter the direct commands:

```
LET A=1
GOTO 10
```

When the 0/40 appears at the bottom of the display, load the preliminary TE module again from your homework tape using the command:

```
LOAD "TE"
```

When the prompt appears on the screen and the cursor appears, type in 62 characters this time before pressing ENTER. We use 62 characters this time to allow for the typist making an error by over-running the end of line marker (the left hand quote symbol). Type the keyword STOP (shifted A key) and press ENTER. When the new prompt appears, press any key. When the diagonal load lines appear on the display, press BREAK. Enter the direct commands:

```
CLEAR
PRINT PEEK 17172
```

The value returned should be 255. Continue PEEKing the addresses down in memory until you locate where the series breaks. All going properly it should be at address 17039 with the value returned of 186. This address marks the highest address that the top of the Calculator Stack used during the program run. The Lower Limit of the Safe Area is the next higher address 17040 with the value returned of 123. You must be careful not to do anything that would LIST the program during the determination of the Lower Limit as this would mess up the values and you would have to start over.

We now have the data we need to find out the number of bytes in the Safe Area of the TE module (each address holds one byte or eight bits). The formula for this is:

```
Upper Limit-Lower Limit+1
```

For the TE Module:

```
17351-17040+1=312
```

As there must always be 36 bytes in the Spare Area of

the memory to avoid an out of memory error, the approximate lowest value to which we can set RAMTOP and have the program and the computer operate properly is:

17408-312+36=17132 (APPROXIMATE)

The value is approximate because when the top of the lculator Stack is at its maximum address, the Machine ack may extend down a minimum number of addresses below RAMTOP or vice versa. Because of this some programs will operate without adding any or all of the 36 bytes. Trial and error is the way to find out. In this case we do not have to add any of the 36 bytes and the minimum address to which RAMTOP can be set for the final version of the TE module is:

17408-312=17096

The typist can actually overrun 31 characters and press ENTER without the computer acting up. As it is standard programming procedure never to store any data in the address of RAMTOP, in line 180 of the final version of the TE module and line 140 of the final version of the PRT module, we use 17097.

The value of the variable B used in both the TE and PRT modules is dependent upon the amount of RAM that you have. As an example, let us compute the value of B for a ZX81 with only 1k RAM. The first nonexistent address is 17408. The number of characters that could be stored would be:

17407-17097+1=311

The number of 32 character lines that can be stored are:

INT (311/32)=9

The number of characters in nine 32 character lines are:

9*32=288

When we are out of space to store a complete 32 character line, we want to stop the text input or the printer and in the TE module, a STOP character code (227) stored at:

B=17097+288=17385

As "repetition is the key to learning" and because each module presents a slightly different problem, let's termine the addresses of the UL, E_Line, LL, and umber of bytes in the Safe Area for the other two modules, starting with PRT.

Clean the memory by turning off the computer. In order to have the printer stop after printing one blank 32 character line, we need to POKE the address:

17409+32=17441

with the code for the BASIC keyword STOP (227). Also we want the fast mode and to set RAMTOP to 17408 prior to loading the preliminary PRT module from the homework tape. Enter the direct commands:

```
FAST
POKE 17441,227
POKE 16389,68
NEW
LOAD "PRT"
```

When the PRINT TEXT query appears, energize the TS2040 printer and press the on switch. Type in Y and press ENTER. After one blank line is printed and the query appears again, type in N and press ENTER. When the diagonal load lines appear, press BREAK.

From this point on proceed as before to locate the Upper Limit. The last value prior to nothing but zeros will be a 224 in address 17351 and the UL will be 17349.

Find the address of E_Line for the PRT module by entering the same direct command used for the TE Module. The value returned should be 16859 which will be used in line 10 of the Flypaper program.

To determine the Lower Limit of the Safe Area of the preliminary PRT module, clean the memory by entering NEW. Type in the program of Figure No.5, changing line 10 to read:

```
10 FOR N=16859 TO 17113
```

ter the direct command:

```
LET A=1
GOTO 10
```

When 0/40 appears, enter:

```
LOAD "PRT"
```

```
FLYPAPER PROGRAM

10 FOR N=16918 TO 17172
20 POKE N,A
30 LET A=A+1
40 NEXT N
```

FIGURE NO. 5

Operate the program as before. After using the BREAK key,key, enter the direct command:

```
CLEAR
```

This wipes out any reserved space, variables, or strings stored in the VARS area, moving all the areas above it up to the top of the Calculator Stack, down in memory, so that PEEKing the addresses will not write over where the top of the Calculator Stack was during the program run. Locate where the series breaks by PEEKing the addresses down in memory from 17113. You should find this at 16911 with a zero returned. The Lower Limit would be address 16912 with a 54 returned.

The number of bytes in the Safe Area are:

UL-LL+1=17349-16912+1=438

The approximate minimum address for RAMTOP for the PRT module would be:

17408-438+36=17006

By trial and error RAMTOP could be set at 17003 and the PRT module and the computer would operate properly.

If your computer is still on with RAMTOP at 17408, enter NEW to clean the memory. Otherwise POKE 16389 with 68 and use NEW. Then load the preliminary RT module from your homework tape. Proceeding in a similar manner as you did for other modules, find the address of the UL and E_Line which should be 17347 and 16765 respectively.

The program in the RT module does not use the VARS area. Therefore CLEAR will not move the top of the Calculator Stack down in memory so that peeking the addresses to find the Lower Limit would write over where it was during the program run. To fix this problem enter the direct command:

```
DIM A$(64)
```

This reserves space in the VARS area for the string A$ and moving the areas above it up in memory including the Calculator Stack. Again find the address of E_Line. It should now be 16835. Re-record the module using GOTO 10.

Clean the memory using NEW and type in the Flypaper program, changing line 10 to read:

```
10 FOR N=16835 TO 17089
```

Enter the direct commands:

```
LET A=1
GOTO 10
```

After the 0/40 appears, enter:

```
LOAD "RT"
```

Use the re-recorded RT module. From this point on proceed in a similar fashion as you did for the other modules to find the Lower Limit. The series breaks at 16870 with a zero returned. The address above is 16871 with a 37 returned. Adjusting this address for the space reserved in VARS, the Lower Limit = 16871-70=16801.

The number of bytes in the Safe Area of the preliminary RT module is:

UL-LL+1=17347-16801+1=547

The approximate minimum address to which RAMTOP can be set for the RT module is:

17408 547+36=16897

As the highest minimum address to which RAMTOP can be set for any of the modules in the chained program is 17096, then this is the value that must be used for the final versions of the modules.

To coin a phrase...explore and "master the possibilities" of your computer. With properly designed software and hardware add-ons, there is nothing a Big Blue, Apple, or AT&T can do that you cannot.

Beginning Z80 Machine Code

LESSON SEVEN

By Syd Wyncoop

Editor: Syd Wyncoop has contributed an excellent article called "A STUDY IN NUMBERS". Due to space limitations in this issue, we will run it next time. The article discusses many of the number systems (bases) that are used by computer programmers, such as Decimal, Binary, and Hexadecimal. Hopefully, this will bring in to focus and act as a compendium for students of our "machine code class", but also should be of interest to all.

Before we begin, I need to ask again for some feedback from you. Especially if you are a TS1000 user. I have heard from no TS1000 users and will concentrate the programming on the 2068 if you don t speak up! The MC instructions are the same for both computers however, each program must be tailored to the operating system. This makes writing this series more difficult. Also, I need your ideas. What would you like to see? We are near done with Z80 instructions.

Let's talk about the logical instructions, And, Or and Xor. And and Or do not give the true/false response you are familiar with if you have used them as Basic boolean operators. Instead, they and Xor operate on the individual bits of a register, or other 8 bit location. Also, the flags are always affected according to the result of these instructions.

Chart 10 provides the truth tables that explain how each of the logical instructions affects the bits being operated upon. While this makes the individual operations clear, it does little to help you understand the instruction. And F0h, when it is encountered. In order to understand these instructions better, it is necessary to understand a little of Base 2 (binary) numbers.

Hex/Bin Conversions

Hex	Bin	Hex	Bin
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Truth Tables

And			Or			Xor		
	0	1		0	1		0	1
0	0	0	0	0	1	0	0	1
1	0	1	1	1	1	1	1	0

As we discovered in our discussion of hex numbers, the highest digit in any base is equal to base-1. This means that we have 2 digits in binary, 0 and 1. The typical number 240 or F0h is 11110000b in binary. The b denotes a binary number just as our h means hex.

Rather than provide a full decimal to binary conversion chart, I have given you a hex to binary chart. This is because we have been working with hex numbers which are a very good shorthand for binary. Eight digit binary numbers are very easily represented by two digit hex numbers. I have provided program 1 for those of you wishing to generate your own charts.

I think you will agree that all those 1's and 0's of binary are begging for us to make an error. That being true, why do we want to represent numbers in binary? The reason is because the logical instructions operate on individual bits and these bits can be easily represented as set or reset (on or off, if you prefer) which is 1 or 0, respectively. Binary provides an easy way for us humans to determine how our friend CPU will act.

Let s look at And. And is often used to mask off unwanted bits. Suppose our routine puts the result in the accumulator and we want to insure that the result is never greater than 7. We would do this with the instruction, And 07h. If A contains 52h, the And 07h would make A contain 02h.

```
A = 01010010 = 52h
And 00000111 = 07h
A = 00000010 = 02h
```

The result is always placed back in the A register. We have effectively said we only want to know about the three least significant bits of A, therefore we have discarded the rest.

Or is used to set the bits we need. If we wanted to insure the most significant bit is set we would Or 80h. If we wanted to insure the most significant bit is reset, we would And 7Fh. Can you see where 10000000b and 01111111b are more useful than 80h and 7Fh with these instructions? Binary allows you to see exactly what is happening.

```
A = 01010010 = 52h          A = 01010010 = 52h
Or  10000000 = 80h          And 01111111 = 7Fh
A = 11010010 = D2h          A = 01010010 = 52h
```

Chart 11

Logical	Rotate and Shift
And r	Rlca
And n	Rla
And (HL)	Rrca
Or r	Rra
Or n	Rlc r
Or (HL)	Rlc (HL)
Xor r	Rl r
Xor n	Rl (HL)
Xor (HL)	Rrc r
	Rrc (HL)
	Rr r
Bit Manipulation	Rr (HL)
Bit b,r	Sla r
Bit b,(HL)	Sla (HL)
Set b,r	Sra r
Set b,(HL)	Sra (HL)
Res b,r	Srl r
Res b,(HL)	Srl (HL)
	Rld
	Rrd

An example of using Or would be when we want to calculate an address. We would calculate the offset in A and then Or it with the high byte of the address to complete the calculation.

Xor is not a fugutive from the Outer-Limits! It is a special case that sets only those bits that differ. For example:

```
A = 10010110 = 96h
Xor 01011101 = 5Dh
A = 11001011 = CBh
```

This is referred to as complimenting. Xor is complicated and is not used often but it is handy at times.

The bit manipulation instructions are the largest single group of Z80 instructions. They are Set, Res and Bit. They are easily understood as they set, reset or test the status of any bit in any register or address.

Set and Res are the set and reset instructions respectively and they do not affect any flags. These instructions are useful when you need to set/reset a bit without affecting the other bits in that byte. You could use And and Or to accomplish the same task but often you will not know the status of the other bits. Set and Res avoid this problem.

Bit is the test instruction. The bits are unchanged but the zero flag is used to indicate the results of the test. The flag is set if the tested bit is zero, and reset if the bit is one.

The rotate and shift instructions are also bit manipulation instructions. They are classified separately as they operate on the entire byte. Many of them use the Carry flag to store a bit.

Rlca rotates the contents of the accumulator left one bit, placing the sign (most significant) bit in Carry as well as in bit 0. The effect of this instruction is to multiply A by 2. For example:

11001000b becomes 10010001b

Graphically it looks like:

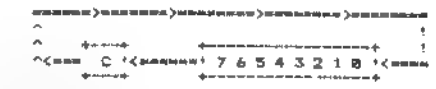
Program 1

```
 10 LPrint "Dec  Hex  Bin"
 20 For i=0 to 255
100 Let h$="  "
110 Let h1=Int (i/16)
120 Let h$(1)=Chr$ (h1+48+(7
And h1>9))
130 Let h2=i-h1*256
140 Let h$(2)=Chr$ (h2+48+(7
And h2>9))
200 Let b$="00000000"
210 Let a=i
220 For n=7 to 0 Step -1
230 If (a-2*Int (a/2)) Then Let
b$(n+1)="1"
240 Let a=Int (a/2)
250 Next n
300 Let t=(i And i<10)+(i And
i<100)+(0 And i<1000)
310 LPrint Tab t;i;Tab 5;h$;Tab
10;b$
320 Next i
```

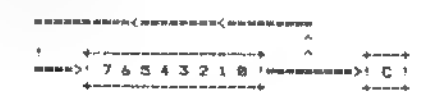
TS1000 users need to change the following lines:

```
120 Let h$(1)=Chr$(h1+28)
140 Let h$(2)=Chr$(h2+28)
```

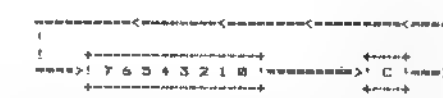
Rla rotates the accumulator bits left though Carry. The Carry flag still contains bit 7 but the prior Carry flag is now in bit 0. Here it is graphically:



Rrca is similiar to Rlca. This time the accumulator's bits are rotated right. Bit 0 is copied into Carry and bit 7. The effect of this instruction is to divide A by 2. This one looks like:

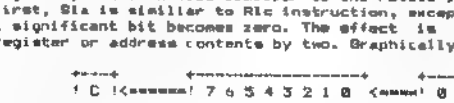


Rra is not surprisingly like Rla, except that we are rotating right. Bit 0 is rotated through the Carry flag. Here it is:

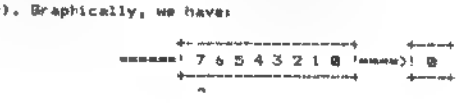


The remaining rotate instructions will operate on any register (including A) or the contents of any address. Rlc r is graphically the same as Rlca. Rl r is graphically the same as Rla. Rrc r is graphically the same as Rrca. Rr r is graphically the same as Rra. The difference between Rra and Rr A is that Rra affects only the Carry flag while Rr A affects all the flags.

The shift instructions are the true arithmetic instructions although they are otherwise similiar to the rotate instructions. The first, Sla is similiar to Rlc instruction, except that the least significant bit becomes zero. The effect is to multiply the register or address contents by two. Graphically we have:



Sra will shift right arithmetic the bits in the specified register or address. This is similiar to Rrc except that bit 0 is only copied to the Carry flag. Bit 7 remains as it was. The effect of this is to divide signed numbers by two, leaving the carry set if there was a remainder (you were dividing an odd number). Graphically, we have:



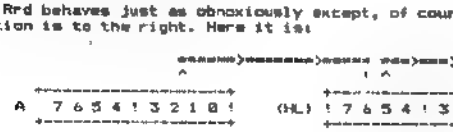
Srl, or shift right logical is the same as Sra, except that the most significant bit becomes zero. Graphically, this is the reverse of Sla:

The last two shift instructions, I have never found a use for. They are Rld and Rrd, which mean rotate left decimal and rotate right decimal, respectively. They operate on the contents of the memory location addressed by HL and the accumulator.

In the case of Rld (not to be confused with Rl d) the low nybble of (HL, is copied into the high nybble of (HL), the high nybble of (HL) is copied into the low nybble of the accumulator, and the low nybble of the accumulator is copied into the low nybble of (HL). Got it? Here's a picture:

For example, assume A contains 7Ah and (HL) contains 31h. After an Rld instruction, A will contain 73h and (HL) will contain 1A.

Rrd behaves just as obnoxiously except, of course, that the rotation is to the right. Here it is:



Remember, you will be limited to an eight bit answer with these instructions. The Carry flag will indicate an overflow and the accumulator, register or memory location will contain the difference. In other words, all arithmetic results will be modulo 256.

Now, how about a practical application? Let's develope a hex dump routine. We can show any byte as two hex digits once we know where to begin. We need the Basic interface first:

```
Hex Dump Interface

  10 Print "Dump from decimal
address: ";
  20 Input a
  30 Print a
  40 Print
  50 Poke base-1,int (a/256)
  60 Poke base-2,a-int
(a/256)*256
  70 Rand Usr base
  75 Rem base=address of Hex
Dump,substitute your addresses
for base
  80 If Inkey$="" Then Goto 80
  90 If Code Inkey$=13 Then Goto
70
  95 Rem 13=Enter on the TS2068
     Use 118 on the TS1000
 100 If Inkey$="Z" Then Copy
 110 Goto 80
```

And now the Hex Dump routine for the TS2068. Remember to use your addresses in place of the xxxx.

```
          Store   Equ HexDmp-2
FDCB023A  HexDmp  Res 0,(TVFlag)   ;this bit tells the Rom
                                   ;routine to print in the
                                   ;main screen area
2Axxxx            Ld HL,(Store)    ;get address to begin dump
0E10              Ld C,10h         ;counter=0 of lines to dump
7C        OutrLp  Ld A,H           ;get the high byte of the
                                   ;first address in this line
                                   ;of the dumped bytes
CDxxxx            Call HexPrt      ;go print it
7D                Ld A,L           ;get low byte of address
CDxxxx            Call HexPrt      ;go print it
3E20              Ld A,20h         ;ascii space
F5                Push AF          ;save the space character
D7                Rst 10h          ;print the space
F1                Pop AF           ;retrieve the character
D7                Rst 10h          ;print another space
0608              Ld B,08h         ;counter=8 bytes/line
7E        InnrLp  Ld A,(HL)        ;byte to accumulator
CDxxxx            Call HexPrt      ;go print it
3E20              Ld A,20h         ;ascii space
D7                Rst 10h          ;print the space
23                Inc HL           ;advance byte pointer
10F6              Djnz,InnrLp      ;loop for 8 bytes
3E0D              Ld A,0Dh         ;ascii carriage return
D7                Rst 10h          ;go to start of next line
0D                Dec C            ;count one line done and
20E0              Jr nz,OutrLp     ;loop for 16 lines
22xxxx            Ld (Store),HL    ;store start of next line
```

```
C9          Done     Ret                    ;return to basic

F5          HexPrt   Push AF                ;save it
E6F0                 And F0h                ;mask off high nybble
1F                   Rra                    ;and rotate to low nybble
1F                   Rra
1F                   Rra
1F                   Rra
CDxxxx               Call Print             ;go print digit in A
F1                   Pop AF                 ;retrieve it
E60F                 And 0Fh                ;mask off low nybble
CDxxxx               Call Print             ;go print digit in A
C9                   Ret                    ;ret to calling routine

FE0A        Print    Cp 0Ah                 ;check if digit is greater
                                            ;than 9
3F                   Ccf                    ;if so, set carry and
DCxxxx               Call c,Offset          ;go adjust for correct
                                            ;ascii character codes
C630                 Add A,30h              ;make a printable char code
E5                   Push HL                ;save registers
C5                   Push BC
D7                   Rst 10h                ;Rom print routine
C1                   Pop BC                 ;restore registers
E1                   Pop HL
C9                   Ret                    ;return to calling routine

C607        Offset   Add A,07h              ;adjust digit to skip over
                                            ;ascii characters between
                                            ;9 and A
C9                   Ret                    ;return to calling routine
```

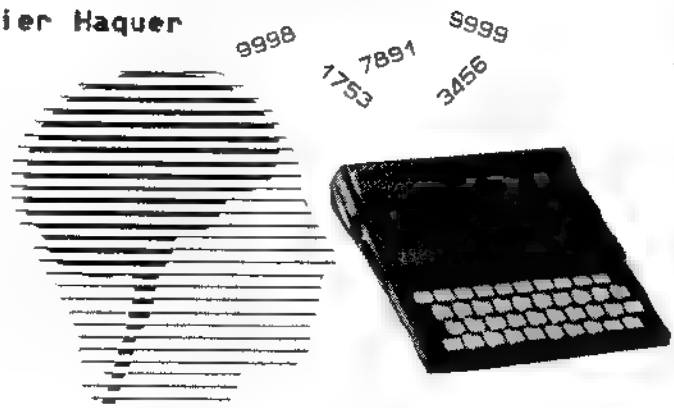
And for you TS1000 owners (I still use mine!):

```
2A7B40      HexDmp   Ld HL,(Store)          ;get address to begin dump
0E10                 Ld C,10h               ;counter-#of lines to dump
7C          Outrlp   Ld A,H                 ;get the high byte of the
                                            ;first address in this line
                                            ;of the dumped bytes
CDA940               Call HexPrt            ;go print it
7D                   Ld A,L                 ;get low byte of address
CDA940               Call HexPrt            ;go print it
AF                   Xor A                  ;same as Ld A,00h
F5                   Push AF                ;save the space character
D7                   Rst 10h                ;print the space
F1                   Pop AF                 ;retrieve the character
D7                   Rst 10h                ;print another space
0608                 Ld B,08h               ;counter-# bytes/line
7E          Innrlp   Ld A,(HL)              ;byte to accumulator
CDA940               Call HexPrt            ;go print it
AF                   Xor A                  ;get space character
D7                   Rst 10h                ;print the space
23                   Inc HL                 ;advance byte pointer
10F7                 Djnz,Innrlp            ;loop for 8 bytes
3E76                 Ld A,76h               ;get carriage return char
D7                   Rst 10h                ;go to start of next line

0D                   Dec C                  ;count one line done and
20E2                 Jr nz,Outrlp           ;loop for 16 lines
227B40               Ld (Store),HL          ;store start of next line
C9          Done     Ret                    ;return to basic

F5          HexPrt   Push AF                ;save byte
E6F0                 And F0h                ;mask off high nybble and
1F                   Rra                    ;rotate to low nybble
1F                   Rra
1F                   Rra
1F                   Rra
CDBA40               Call Print             ;go print digit in A
F1                   Pop AF                 ;retrieve byte
E60F                 And 0Fh                ;mask low nybble
CDBA40               Call Print             ;go print it
C9                   Ret                    ;return to calling routine

C61C        Print    Add A,1Ch              ;make a character 0 to F
E5                   Push HL                ;save registers
C5                   Push BC
D7                   Rst 10h                ;Rom print routine
C1                   Pop BC                 ;restore registers
E1                   Pop HL
C9                   Ret                    ;return to calling routine
```

I have to assume that if you are still with me, you have obtained some good study aids. Since almost all books on the subject of Z80 MC have numerous tables in them, this is the last time I will give the hex codes in the MC disassemblies. I will instead, provide the source code. What's source code? That's another lesson. See ya soon!

NUMBER MADNESS

by Zack Xavier Haquer



Here is a challenging game for those of you who are tired of shoot-em-ups and pac-persons. The computer "thinks" of a four digit number, and your job is to guess (or rather, deduce) the number. Each time you guess, the computer gives you a clue which will tell you how many digits are correct and in the right place (A), correct but in the wrong place (B), or completely wrong (C). Note that the order of the letters in the clue DOES NOT necessarily correspond with the position of the digits!

You have the option of selecting five levels of play. At the hardest level, only your current guess is displayed, making it a challenge to memory as well as logic.

Ten "sets" constitute a game. The average number of guesses it takes is your score; the lower, the better. (A maximum of 15 guesses are allowed.) After a game, your name is entered into a "Hall of Fame", which can then be saved to tape (along with the program and other variables).

Being entirely in ZX BASIC, with no "tricks," this program can be easily converted to TS2068 or Spectrum. The changes are as follows:

1: Remove FAST and SLOW commands.

2: Replace referenes to CHR$ 118 to CHR$ 13 (as in the IF INKEY$=CHR$ 118 THEN... lines).

3: Replace dummy FOR-NEXT loops with PAUSE (the FOR-NEXT was used instead of PAUSE to prevent the blink that results from using PAUSE with the ZX81). For example, delete lines 300 and 305 and use 300 PAUSE 60 instead.

4: Modify the SAVE command at 2590 as desired. For example, you might want to use:

2590 SAVE "scr" SCREEN$: SAVE "guess" LINE 3000

To load the program and screen, use LOAD "scr" CODE: LOAD "guess"

5: Add color, FLASH, sound (BEEP), and lower-case as desired.

```
   1 REM GUESS MY NUMBER
   5 SLOW
  10 DIM A(8)
  15 DIM B(5)
  17 DIM C(5)
  20 DIM Z$(1,9)
  25 DIM C$(5,9)
  30 CLS
  50 LET J=0
  60 LET K=0
 100 FOR A=6 TO 21 STEP 5
 110 PRINT AT 8,A;"█████"
 120 NEXT A
 130 FOR A=9 TO 11
 140 PRINT AT A,25,"█"
 150 NEXT A
 160 FOR A=21 TO 6 STEP -5
 170 PRINT AT 12,A,"█████"
 180 NEXT A
 190 FOR A=11 TO 9 STEP -1
 200 PRINT AT A,6,"█"
 210 NEXT A
 220 FOR A=7 TO 22 STEP 3
 230 PRINT AT 9,A,"▒▒▒"
 240 NEXT A
 250 PRINT AT 10,24;"▒"
 260 FOR A=22 TO 7 STEP -3
 270 PRINT AT 11,A,"▒▒▒"
 280 NEXT A
 290 PRINT AT 10,7,"GUESS MY NUMBER"
 292 RAND
```

```
 295 PRINT AT 3,5;INT (10000*RND
),"?";AT 5,27;INT (10000*RND),"?
",AT 15,3;INT (10000*RND),"?"
 300 FOR A=1 TO 15
 305 NEXT A
 310 PRINT AT 21,0;"[illegible]
[illegible] PRESS  ENTER  "
     PRINT AT 20,0,"SELECT DIFFI
CULTY 1 - 5."
 330 LET A$=INKEY$
 340 IF A$)"0" AND A$<"5" THEN G
OTO 400
 350 IF A$=CHR$ 118 THEN GOTO 50
00
 360 GOTO 330
 400 CLS
 405 LET T=VAL A$
 410 PRINT "HELLO, THERE."
 420 PRINT AT 3,5,"WHATS YOUR NA
ME?";AT 20,0;"(TYPE NAME, UP TO
8 CHARACTERS,  THEN ENTER.)"
 430 INPUT Z$(1)
 440 LET A=8
 450 IF Z$(1,A)<>" " THEN GOTO 4
90
 460 LET A=A-1
 470 IF NOT A THEN GOTO 400
 480 GOTO 450
 490 LET Z$(1,A+1)="."
 500 CLS
 505 LET L=0
 510 PRINT "OK, ",Z$(1),"HERE GO
ES."
 520 FOR A=1 TO 20
 525 NEXT A
 530 CLS
 540 LET J=J+1
 550 PRINT AT 0,20 "GAME NO. ",J
 560 LET E=INT (10000*RND+500+(L
+J))
 570 IF E<=9999 THEN GOTO 600
 580 LET E=E-INT (5000*RND)
 590 GOTO 570
 600 LET A=E
 610 FOR B=4 TO 1 STEP -1
 620 GOSUB 9000
 630 NEXT B
 635 PRINT AT 1,8 "TRY NUMBER CL
LE"
 640 LET L=L+1
 645 PRINT AT L+2,8,"[illegible]
[illegible]"
 650 IF L<4 AND J=1 THEN PRINT A
T 21,0;"(TYPE NUMBER, THEN ENTER
)"
 660 INPUT F
 665 IF F<=9999 THEN GOTO 680
 670 PRINT AT L+2,8,"TOO HIGH,
,Z$(1)
 672 FOR A=1 TO 15
 675 NEXT A
 677 GOTO 645
 680 FAST
 685 PRINT AT L+2,8;'
"
 690 IF L<4 AND J=1 THEN PRINT A
T 21,0;"
"
 710 PRINT AT L+2,9,L;AT L+2,13,
"0000";AT L+2,17-LEN STR$ F,F
 715 IF E=F THEN GOTO 1200
 720 LET A=F
 730 LET H=0
 740 LET I=0
 750 FOR B=8 TO 5 STEP -1
 760 GOSUB 9000
 770 NEXT B
 780 FOR B=5 TO 8
 790 FOR C=1 TO 4
 800 IF A(C)=A(B) THEN GOTO 820
 810 GOTO 840
 820 LET H=H+1
 830 LET C=4
 840 NEXT C
 850 NEXT B
 860 FOR B=5 TO 8
 870 LET C=B-4
 880 IF A(C)=A(B) THEN GOTO 900
 890 GOTO 920
 900 LET I=I+1
 910 LET H=H-1
 920 NEXT B
 930 LET V=I+10+H*2+9100
 940 GOSUB V
 950 LET U=6-T
 960 IF U=5 THEN LET U=15
 970 LET U=L-J
 980 IF U<1 THEN GOTO 1000
 990 PRINT AT U+2,19,"
  00 PRINT AT L+2,19,V$
1005 SLOW
1010 IF L<=14 THEN GOTO 640
1020 PRINT AT 18,5;"YOU BLEW IT;
";Z$(1)
1021 PRINT
1022 PRINT TAB 5,"THE NUMBER IS
",E

1030 GOTO 1250
1050 GOTO 1040
1200 PRINT AT L+2,19;"AAAA",
1201 SLOW
1202 FOR A=1 TO 3
1203 FOR B=1 TO 5
1204 NEXT B
1205 PRINT "H";
1206 NEXT A
1207 PRINT ","
1210 PRINT AT 18,3;"CONGRATULATI
ONS, ";Z$(1);AT 20,5;"***YOU GUE
SSED IT***"
1250 LET K=K+L
1260 LET L=0
1270 PRINT "(PRESS ""ENTER"" TO
CONTINUE.)"
1280 IF INKEY$=CHR$ 118 THEN GOT
O 1300
1290 GOTO 1280
1300 CLS
1305 IF J=10 THEN GOTO 2000
1310 PRINT "AVERAGE= ";(STR$ (K/
J)+"    ")( TO 4)
1320 GOTO 540
2000 CLS
2010 PRINT AT 2,10;"END OF GAME.
"
2020 PRINT
2025 LET K=K/10
2030 PRINT TAB 3,"YOUR 10-GAME S
CORE IS ",K,
2035 IF K-INT K=0 THEN PRINT ".0
"
2050 LET A$="********************
***"
2060 LET B$="**
  **"
2070 PRINT AT 6,5,A$
2080 PRINT AT 7,5;"*[illegible]
[illegible]*"
2090 PRINT AT 8,5,A$
2100 PRINT AT 9,[illegible];"** NAME      A
VG LVL **"
2110 FOR A=10 TO 15
2120 PRINT AT A,5,B$
2125 NEXT A
2130 PRINT AT 16,5,A$;AT 17,5,A$
2190 LET X=1
2200 IF B(X)=0 THEN GOTO 2305
2205 IF K>B(X) THEN GOTO 2210
2207 GOTO 2240
2210 LET X=X+1
2220 IF X=6 THEN GOTO 2490
2230 GOTO 2200
2240 FOR A=5 TO X+1 STEP -1
2250 LET B(A)=B(A-1)
2255 LET C(A)=C(A-1)
2260 LET C$(A)=C$(A-1)
2265 NEXT A
2305 LET C$(X)=Z$(1)
2306 LET B=9
2307 IF C$(X,B)=" " THEN GOTO 23
10
2308 IF C$(X,B)="." THEN LET C$(
X,B)=" "
2309 GOTO 2315
2310 LET B=B-1
2312 GOTO 2307
2315 LET B(X)=K
2316 LET C(X)=T
2317 PRINT
2318 LET B=9
2319 LET B$=C$(X)( TO B)
2320 IF B$(B)=" " THEN GOTO 2322
2321 GOTO 2325
2322 LET B=B-1
2323 GOTO 2319
2325 PRINT "***CONGRATULATIONS**
*",B$,"***"
2330 LET J=1
2340 GOTO 2500
2490 LET U=0
2500 PRINT AT 10,8,"--------- ---
----"
2505 FOR A=1 TO 5
2510 PRINT AT 10+A,8 C$(A),AT 10
+A,17;B(A);AT 10+A,22,C(A)
2520 NEXT A
2525 IF X=6 THEN PRINT AT 19,0 "
TOO BAD,",Z$(1)," "K, = NO CIGA
R."
2530 IF U=1 THEN GOTO 2550
2540 GOTO 3010
2550 PRINT AT 20,4,"[illegible]
[illegible]"
2560 PRINT "REWIND TAPE - RECORD
 - "'ENTER"","
2565 POKE 16418,0
2567 PRINT "BYPASS - PRESS "'B""
"
2570 IF INKEY$=CHR$ 118 THEN GOT
O 2590

2575 IF INKEY$="B" THEN GOTO 300
0
2580 GOTO 2570
2590 SAVE "GUESS"
3000 PRINT AT 20,4,"
                "
3010 PRINT AT 21,0,"  PRESS  "P'
" TO START NEW GAME."."
                "
3015 POKE 16418,2
3020 IF INKEY$="P" THEN GOTO 30
3030 GOTO 3020
4000 CLEAR
4010 SAVE "GUESS"
4020 RUN
5000 CLS
5010 PRINT TAB 10;"[illegible]"
5020 PRINT
5030 PRINT TAB 5,"WE WILL PLAY 1
0 ""ROUNDS"" OFA GAME IN WHICH I
 (THE COMPUTER)THINK OF A 4-DIGI
T NUMBER AND   YOU TRY TO GUESS
WHAT IT IS."
5050 PRINT TAB 5;"EVERY TIME YOU
 GUESS,I WILLGIVE YOU A CLUE WHI
CH WILL HELP YOU GUESS MY NUMBER
."
5060 PRINT
5070 PRINT TAB 5;"FOR EXAMPLE,IF
 MY NUMBER IS","?=31"
5080 PRINT "AND YOJ GUESS:","351
4    THEN THE"
5090 PRINT "CLUE WILL BE ","ABBC
"
5091 PRINT
5093 PRINT "A- RIGHT DIGIT,RIGHT
 PLACE (5)"
5094 PRINT "B- RIGHT DIGIT,URONG
 PLACE (1,3)"
5095 PRINT "C- WRONG DIGIT
      (4)"
5100 PRINT
5110 PRINT "AVERAGE NUMBER OF TR
IES = SCORE."
5120 PRINT
5130 PRINT "  (PRESS ""ENTER"" T
O CONTINUE,)"
5140 IF INKEY$=CHR$ 118 THEN GOT
O 5160
5150 GOTO 5140
5160 CLS
5170 PRINT TAB 5,"[illegible]
[illegible]",,,
5190 PRINT TAB 5,"YOU HAVE THE O
PTION OF FIVE LEVELS OF DIFFICULT
Y (1 - 5)."
5200 PRINT
5210 PRINT "1 - EASY, YOU SEE AL
L YOUR ""OLD""    CLUES."
5220 PRINT
5230 PRINT "2 - YOU SEE CLUES FO
R LAST FOUR     GUESSES."
5240 PRINT
5250 PRINT "3 - YOU SEE CLUES FO
R LAST THREE    GUESSES."
5260 PRINT
5270 PRINT "4 - YOU SEE CLUES FO
R LAST TWO      GUESSES."
5280 PRINT
5290 PRINT "5 - MOST DIFFICULT.
YOU SEE CLUE    FOR PRESENT GUES
S ONLY."
5300 GOTO 320
9000 LET C=INT (A/10)
9010 LET A(B)=A-C*10
9030 LET A=C
9050 RETURN
9100 LET V$="CCCC"
9101 RETURN
9102 LET V$="BCCC"
9103 RETURN
9104 LET V$="BBCC"
9105 RETURN
9106 LET V$="BBBC"
9107 RETURN
9108 LET V$="BBBB"
9109 RETURN
9110 LET V$="ACCC"
9111 RETURN
9112 LET V$="ABCC"
9113 RETURN
9114 LET V$="ABBC"
9115 RETURN
9116 LET V$="ABBB"
9117 RETURN
9120 LET V$="AACC"
9121 RETURN
9122 LET V$="AABC"
     RETURN
9124 LET V$="AABB"
9125 RETURN
9130 LET V$="AAAC"
9131 RETURN
9132 LET V$="AAAB"
9133 RETURN
```

Still using only 16K of TS1000 RAM?

Now cheaply double or triple your memory and unleash your computer's true power.

Zebra Systems has purchased a large lot of new Memotech 16K and 32K RAMs directly from England so that we could offer them at drastically reduced prices.

Memotech RAMs are the highest quality and best-selling expansion RAMs for Timex/Sinclair computers. They have sleek looking black extruded-aluminum cases that conform to the shape of your TS1000 or ZX81. These 16K and 32K RAMs feature feed-through connectors and circuitry that allows you to use them along with the 16K RAMs you already have, so that you can acheive respectively, 32K or 48K total system RAM capacity.

Each RAM comes with an easy to understand instruction manual explaining how to use the RAM to best advantage, and a 15-day money-back guarantee from Zebra Systems. Order yours now, while the supply lasts.

Acheive 32K with Memotech 16K + your Timex 16K

or

Acheive 48K with Memotech 32K + your Timex 16K

CAT# R416 Memotech 16K RAM ... $14.95

CAT# R432 Memotech 32K RAM ... $24.95

CAT#S116 Sinclair 16K RAM, TS1016 equivalent... $10.00

Include $3.00 S&H VISA/MC accepted. NY residents add sales tax.

ZEBRA SYSTEMS, INC.

78-06 JAMAICA AVE., WOODHAVEN, NY 11421, (718) 296-2385

SPRING SALE

- ROMSWITCH 2--Improved! (CMOS) $39.95
  Corrected TS2068 ROM + Spectrum ROM
- TS2068 + ROMSWITCH 2 installed $189.00
- Spectrum ROM $16.95
- Spectrum ULA chip special design for correct display on US color TV/monitor. RARE! $40.00
- Timex 2040 Printer, new w.spacers $49.95
- QL ElectroCAD for 512K,elect layout $29.95

QL, TS2068, TS1000/1500/ZX81 hardware/software; electronic parts, books/magazines, liquidation

Send for free catalog. Free gift w. order.
Send check NO MC/Visa# - $3. shipping on order
814-364-1325 (Usually answering machine)

G. RUSSELL ELECTRONICS

RD 1 • Box 539 • Centre Hall, PA 16828

5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5

You've heard of PROFILE 2068.
You may have even heard of PROFILE +3!
But we are willing to bet that you have never heard of:

PROFILE +5!

That's right PROFILE +5! A great new addition to the line of PROFILE updates. We have used all of the various versions of this program and can honestly say that this is the ULTIMATE PROFILE!

+5 contains all of the features of PROFILE 2068 and +3 as well as all of the new features of TOM WOODS' PROFILE CARTRIDGE! Including the 'NOT' and 'OR' search routines. It also lets you print in 80 columns! It still has the 'NAME REVERSAL' print feature and the 'EDIT A PHRASE' menu option. You can now 'MERGE' files, 'DATE' files and make use of the great machine code sort routine!

All of this is available in several ways, if you already own one or more versions of PROFILE, you will be able to save some money on this version. the complete program on cassette available for:

ONLY $29.95+2ph

(User Manual available only from TOM WOODS for $10.00)
For more information, please call or write:

AVAILABLE EXCLUSIVELY FROM:

RMG ENTERPRISES
1419 1/2 7TH STREET
OREGON CITY, OR 97045
503/655 7484

5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5+5

*** NEW ***
LARKEN
DISK DRIVES

LKDOS - EXTENDED BASIC CARTRIDGE

- Uses all standard (Token) keyboard cassette commands
- CAT MERGE ERASE FORMAT LOAD SAVE PRINT and more
- Uses NO RAM space . HAS 8K ROM and 8K RAM on board
- NMI Memory Save Feature :PUSH BUTTON memory transfer
- Fully 2068 / Spectrum compatible / QS-64 Compatible
- 10 Extended Basic Commands for Graphics,Utilities and up to 3 scrolling Windows on the screen in proportional print (up to 100 characters wide)
- Supports 2 Floppy's and soon to released 256K RAMDISK
- Can be used with either Larken Disk Interface
- Available soon for RAMEX and other disk IF's

NEW 2068/Spectrum Disk Interface
- Supports 1 or 2 - 3" 3.5" or 5.25" SS or DS drives
- Double Density 360k (double sided)
- Snap-shot pushbutton on board

ZX-81 TS1000 2068 Disk Controller
- Supports double sided 5.25" drive , 160K per disk
- The Best DOS for the ZX-81
- Uses no ram :2K Ldos and 2K ram on board
- Connects to Computer with Ribbon Cable

PRICES : (US)	LKDOS EX-BAS Cartridge	$60.00
	NEW 2068 /Spec FD Interface	$45.00
Add $5.00	ZX-81 Disk Controller	$99.00
S&H	256K Ram Disk	(TBA)
	2- Drive Floppy cable	$8.00

* LARKEN ELECTRONICS RR#2 NAVAN ONTARIO CANADA K4B-1H9 *

QL Flight Simulator

REVIEW

Reviewed By Dennis Silvestri

Another flight simulation for a Sinclair computer? What could this program do that the other flight simulation programs cannot do? These were questions I asked before receiving this program. Here are my answers.

The usual view of the instruments and out the cockpit is provided in QL FLIGHT SIMULATOR. All the usual flight controls are present, using either a joystick or the keyboard. The simulation presented is the flying, landing, and taking off of a small single engine aircraft. So far, there is not much difference between this and other programs like it. What makes this program completely different is in its graphic visual displays. There is actual scenery to view such as mountains, valleys, water, power lines, towers, buildings, and other objects. This scenery is shown in what is referred to as hi-res 3D "wire" (or line) graphics. This type of display defines only the outlines and/or contours of the objects they portray. There are also panoramic views out the cockpit. This type of view is relative to aircraft altitude and simulates the actual view you would see if you turned your head left, right, or looked behind you, as well as looking up or down. There are nine different scenic areas to choose from (referred to as "worlds"), which make for some interesting flight simulation. Weather conditions can be set in any of the worlds, which affects wind direction and speed, as well as cloud cover. The weather conditions in each of the nine worlds can be set separate from one another, and can be called up at any time should one forget what they were. Weather readings are also automatically displayed when crossing the boundary from one world to another. An interesting feature occurs with the aircraft on the runway. Here, one can taxi around...pulling up to various objects. There is also a fuel depot which is the only source for refueling the aircraft. A radar view is also provided. This shows the aircrafts position ralative to the ground and any objects. This view is helpful if you are flying above cloud cover.

Since the keyboard is also used to control all functions not related to actual flying, the user should expect to spend considerable time in getting familiar with their functions. No less than 27 keys are used. QL FLIGHT SIMULATOR comes with very extensive documentation which includes maps of the nine worlds. These maps show all the objects and their heights. The maps are very useful, especially when flying between worlds. However, they are poorly printed, and could have been made larger for ease in reading. This program has a built in backup copy feature and uses sound as well as limited color. An RGB monitor is advisable. Using a TV results in the lower lines being cut off from the screen.

There are a number of flight simulation programs available for the Sinclair line (including Spectrum, TS2068, TS1000). The QL FLIGHT SIMULATOR should be considered a leader for programs of this type. As I end this review, I wonder. This program runs on the standard 128k QL. The computer is expandable to 640k. Upgrading this program to the QL RAM limits could result in a very powerful program. QL FLIGHT SIMULATOR should not be looked at as "just a game", but as the first step one could take if interested in learning how to fly.

The QL computer used to review this program was supplied by A+ Computer Response. QL FLIGHT SIMULATOR is available from most QL dealers.

Wind Chill Chart

By Gale Henslee

The following QL program was inspired by an article written by Tom Beatty in the February issue of Computer Shopper. Only the original program's table format and wind chill factor equations are the same, the rest is a complete re-write in QL SuperBASIC. I think the function I used to format the right-justified columns should be of general interest to readers. The procedure "TABLE" outputs the chart to the screen (80 col. monitor) and "CHART" outputs to an Epson compatible, 80 col. printer.

```
100 REMark     ************************************************
110 REMark     *****          WIND CHILL CHART             *****
120 REMark     *****            T. BEATTY                  *****
130 REMark     *****                                       *****
140 REMark     ************************************************
150 REMark     *****       ADAPTED FOR SINCLAIR QL         *****
160 REMark     *****       BY GALE HENSLEE                 *****
170 REMark     *****         AMARILLO, TX                  *****
180 REMark     *****         JANUARY, 1987                 *****
190 REMark     ************************************************
200 REMark MAIN PROGRAM
210 SETUP
220 TABLE
230 CHART
240 STOP
250 DEFine PROCedure SETUP
260 WINDOW #1,512,256,0,0:PAPER #1,0:INK #1,4:CLS #1
270 PRINT #1,"LOADING ARRAY - - - " PRINT #1
280 DIM A(16,14):H1$="#####"
290 PRINT #1,"CALCULATING ROW: ",
300 FOR R=1 TO 16
310 PRINT #1,R;" ";
320 V=2*R+2
330 FOR C=0 TO 13
340 T=-5*C+40
350 A(R,C+1)=((10.45+(6.686112*SQRT(V))-(.447041*V))/22.034*(T-91.4))+91.4
360 NEXT C
370 NEXT R
380 PRINT #1," DONE."
390 PRINT #1:PRINT #1,"NOW PRINTING CHART - - ":CLS #1
400 END DEFine SETUP
410 DEFine PROCedure TABLE
420 F=1
430 AT 0,32 PRINT #F,"WIND CHILL CHART"
440 AT 3,9:PRINT #F,"DIRECTIONS: (1) FIND TEMPERATURE ACROSS TOP ROW (°F)"
450 AT 4,9:PRINT #F,"                (2) FIND WIND VELOCITY (MPH) DOWN LEFT COL
UMN"
460 AT 5,9:PRINT #F,"                (3) INTERSECT ROW AND COL FOR WIND CHILL (°
F)"
470 AT 7,0:PRINT #F,"     ";
480 OLD = 0
490 OLD1 = 5
500 TEMP1% = 10
510 FOR K=0 TO 13
520 TEMP%= -5*K+40
530 AT 7,TAB(TEMP%):PRINT #F,TEMP%;
540 NEXT K
550 OLD1 = 5
560 TEMP1% = 10
570 FOR R=1 TO 16
580 WIND% = 2*R+2
590 AT 7+R,3+GWIND% (10):PRINT #F,WIND%;
600 FOR C=1 TO 14
610 IF A(R,C)< 0 AND A(R,C) > -1.5 THEN GO TO 630
620 TEMP% = INT(A(R,C)+.5):GO TO 640
630 TEMP% = 0
640 AT 7+R,TAB(TEMP%):PRINT #F,TEMP%;
650 NEXT C
660 OLD1 = 5
670 TEMP1% = 10
680 NEXT R
690 END DEFine TABLE
700 DEFine FuNction TAB(TEMP)
710 OLD = OLD1 + 3 * (TEMP% >=10) + 4 * ((TEMP% >=0) AND (TEMP% < 10)) +
3 * ((TEMP% > -10) AND (TEMP% < 0)) + 2 * (TEMP% <= -10)
720 OLD1 = OLD + 2 - ((TEMP% < 10) AND (TEMP% >= 0)) + (TEMP% <= -10)
730 TEMP1% = TEMP%
740 RETurn OLD
750 END DEFine
760 DEFine PROCedure L
770 WINDOW #1,512,208,0,0:CLS #1:CLS #0
780 LIST #1
790 END DEFine L
800 DEFine PROCedure CHART
810 F=5
820 OPEN #5,SER1
830 PRINT #F,TO 32,"WIND CHILL CHART";CHR$(10)
840 PRINT #F,TO 9;"DIRECTIONS: (1) FIND TEMPERATURE ACROSS TOP ROW (°;CHR$(
248),"F)"
850 PRINT #F,TO 9;"                (2) FIND WIND VELOCITY (MPH) DOWN LEFT COLUM
N"
```

```
                    WIND CHILL CHART

     DIRECTIONS: (1) FIND TEMPERATURE ACROSS TOP ROW (°F)
                 (2) FIND WIND VELOCITY (MPH) DOWN LEFT COLUMN
                 (3) INTERSECT ROW AND COL FOR WIND CHILL (°F)

      40   35   30   25   20   15   10    5    0   -5  -10  -15  -20  -25

 4    40   35   30   25   20   15   10    5    0   -5  -10  -15  -20  -25
 6    35   30   24   19   13    8    2   -3   -8  -14  -20  -25  -31  -36
 8    31   25   20   14    8    2   -4  -10  -16  -21  -27  -33  -38  -45
10    28   22   16   10    4   -3   -9  -16  -21  -27  -33  -40  -46  -52
12    26   19   13    6    0   -7  -13  -19  -26  -32  -39  -45  -51  -58
14    23   17   10    3   -3  -10  -17   23   30  -36   43   50   56   63
16    21   15    8    1   -5  -13  -20  -25  -33  -40  -47  -54  -60  -67
18    20   13    6    0   -8  -15  -22  -29  -36  -43  -50  -57  -64  -71
20    18   11    4   -3   10   16   25  -32  -39   46  -53  -60  -67  -75
22    17   10    2   -5   12  -19  -27  -34  -41  -48  -56  -63  -70  -78
24    16    8    1   -6  -14  -21  -28  -36  -43  -51  -58  -65  -73  -80
26    15    7    0   -8  -15  -23  -30  -36  -45  -53  -60  -68  -75  -83
28    14    6    0   -8  -17  -24  -32  -38  -47  -54  -62  -69  -77  -85
30    13    5   -2  -10  -18  -25  -33  -41  -48  -55  -64  -71  -79  -86
32    12    4   -3  -11  -19  -26  -34  -42  -48  -57  -65  -73  -80  -88
34    12    4   -4  -12  -20  -27  -35  -43  -51  -58  -66  -74  -82  -89
```

```
860 PRINT #F,TO 9,"                (3) INTERSECT ROW AND COL FOR WIND CHILL (°;
CHR$(248),"F)";CHR$(10)
870 PRINT #F,"     ";
880 OLD = 0
890 OLD1 = 5
900 TEMP1% = 10
910 FOR K=0 TO 13
920 TEMP%= -5*K+40
930 PRINT #F,TO TAB(TEMP%):TEMP%;
940 NEXT K
950 PRINT #F,CHR$(10)
960 OLD1 = 5
970 TEMP1% = 10
980 FOR R=1 TO 16
990 WIND% = 2*R+2
1000 PRINT #F,TO 3+GWIND% (10):WIND%;
1010 FOR C=1 TO 14
1020 IF A(R,C)< 0 AND A(R,C) > -1.5 THEN GO TO 1040
1030 TEMP% = INT(A(R,C)+.5):GO TO 1050
1040 TEMP% = 0
1050 PRINT #F,TO TAB(TEMP%):TEMP%;
1060 NEXT C
1070 OLD1 = 5
1080 TEMP1% = 10
1090 PRINT #F,CHR$(13)
1100 NEXT R
1110 FOR PLINE = 1 TO 8
1120 PRINT #F,CHR$(13)
1130 NEXT PLINE
1140 CLOSE #5
1150 END DEFine CHART
```

Trouble-shooting the QL

By Gale Hensley

I've now seen a total of six QL's: two assembled at the factory and four kits. I have to say that quality control leaves much to be desired, a situation I hope ill improve. This information is intended to provide help for new or prospective owners, to get started with a little less trouble than I had when a defective computer is received. The two defective parts I've experienced were keyboard leads (1 of 6) and Microdrives (2 of 12). All were replaced under warranty.

The Microdrives can cause you a lot of misery if you plug in your new QL computer and follow the instructions in the User Manual. It tells you to "make at least one backup on a blank cartridge" before you use any QL program. This is good advice, EXCEPT, if one of your new, untested Microdrives is defective, you may destroy the original copy of the program. This happened to me with my first QL, and it took three exchanges of the PSION programs before I (and the dealer) realized that it was the Microdrives causing the problem. You see, Microdrive One worked fine, but Microdrive Two did not. A new tape would load and run fine in MDV1, so I would follow the manual and put a new cartridge in MDV1 and the original cartridge in MDV2 (following the User Manual) and enter LRUN MDV2_CLONE. The new cartridge would format and as soon as MDV2 started verifying the first file copied, the damage was done. The ominous message "At line__ bad or changed medium" would appear and copying would stop. This was hard to believe, since the program had just been loaded and run to make sure it was ok. So, try it in drive 1 again, right? Wrong! .The tape was not readable. Ok, call the dealer and explain what happened...they'll say the QL must be ok because it loaded and ran the tape once and everything looked good...so send the tape back for replacement and try it again. Same results.

Alright, so how do you avoid getting into this situation? It's really easy, you just assume that something will go wrong and go through the following test procedure before trying a tape with anything important on it. If the test is successful, your Microdrives are both in the 10 out of 12 group that are good.

1. Reset the QL and press F1 or F2, as appropriate.
2. Put a blank cartridge in drive 1.
3. Format it with "format mdv1_test1".
4. Enter "dir mdv1_" and you should get a message like:
   test1
   220/221 sectors
5. Write a short program and save it to mdv1. Lrun the program. If it works you can feel confident that mdv1 is a good drive.
6. Repeat the above steps 2 through 5 with mdv2.
7. When you are sure both drives are good, then proceed to make backup copies of your programs. You will probably not have any further trouble.
8. If any of the above steps fail, repeat them to be sure, starting with step 1. If you get the same result, the drive is probably defective. Call your dealer. (If you have a spare and are building a kit, replace the drive and test it. Chances are that the new drive will be ok and everything will work.)

Keyboard leads are simple to troubleshoot. The first thing you should do after turning on the computer is press each printable key and look for the echo in the control area (screen 0). Test the capslock and shift keys and the cursor keys (this checks CTRL). Pressing enter should get you some kind of error message unless you have typed a valid command. If all keys work so far, it's unlikely that there is a bad keyboard lead. If any keys fail to work, it's probably a bad lead, but call your dealer for instructions. A lead that's improperly inserted, or broken near the socket can be fixed easily in a few minutes but don't void your warranty by opening the case (kit owners excepted). A broken lead is usually difficult to spot because the ribbon cables can have hairline cracks that are virtually invisible. If there is a noticeable kink in the ribbon, you should suspect such a crack. You can confirm it with an ohmmeter. A lead broken near the socket can be trimmed off above the bad place and reinserted into the socket. Be careful here, use both hands to hold the ribbon about 1/4 inch above the end and push it into the socket straight above, using gentle but firm pressure. It usually helps to cock the cable to one side to get it started. Be patient here and it will go in. Be careful not to kink the cable.

I hope the above suggestions will help some new owners to get started using their QL's with less hassle.

QL Quill/Word Processor Tips

Part III

by

Mike de Sosa

You may not be aware of it as yet, but Time Designs' advertisers will soon be offering new hardware and software which should, once again, blow the mind of QL Quill users. Think about using your favorite--or least favorite-- word processor program with the following accessories:

1) A dual disk drive interface with an additional 512K RAM; a full CARE ELECTRONICS /QJUMP Toolkit II and the super-efficient, lightning fast QfLash RAMdisk driver and Toolkit and other utilities in EPROM; a parallel printer interface; and an optional MIDI interface and an unusually efficient mouse.

2) A multi-tasking program which permits instant switching between up to 9 machine executable programs (including the four QL software programs, each with reserved data space) and SuperBASIC without loss of data, an on-screen calculator which will printout the results in Quill and the other programs, and a new and unmatched files management program.

3) A revamped keydefine program which offers double-keystroke program operation (41 user-definable function keys, each holding up to 2048 characters of text [about 340 words] or the equivalent in commands) and several utilities such as printing of the date and a CAPS LOCK audible signal in the QL software programs: produce an entire letter with a few keystrokes.

The items described which you will soon see advertised in Time Designs (and which will be fully discussed in my forthcoming book on the Sinclair QL system (Taking the

Quantum Leap! The Last Word on the Sinclair QL, to be published by Time Designs in April) are, in order:

1) One of several available versions of the Sandy SuperQBoard, "the most sophisticated peripheral for the QL," available soon with Supermouse from Sandy (UK) PCP Ltd. [the producers of the Futura 68000 PC] through QUANTUM COMPUTING (an authorized U.S. QL dealer), Box 1280, Dover, NJ 07801 (Tel. [201] 328-8846), price TBA, probably under $500--let's get those prices down, dealers! The other part of this system is a plug-in replacement EPROM for the SuperQBoard by QfLash (the QfLash RAMdisk Driver/Toolkit is also available in six other configurations to meet your needs). Write or call QfLash, Firma Uwe Fisher, Post Box 102121, D-2000 Hamburg 1, West Germany (Tel. 040-7650461) for prices. Tip: While you're at it, order the very excellent QfLash Utilities software--RAMDOCTOR alone is worth the price. And, while we're talking about QfLash products, their latest is another unique bit of hardware: the QfLash EPROM X 12 fits into the QL expansion port and allows up to three EPROMs to be used simultaneously, without switching--the Psion QL software programs may be used all at one time, even on a 128K RAM QL!

2) The multi-tasking program is Taskmaster, available from Sector Software, 39 Wray Crescent, Ulnes Walton, Leyland, Lancashire, U.K. (Tel. 0772 454328), for #10, including Airmail postage. This was termed "the Rolls-Royce of such software" by a British reviewer.

3) The key-define program is the "new and updated" Keydefine, available from many distributors, including Sector Software, for about #10 plus postage. This suite of programs, useful in SuperBASIC and assembly language programming as well as in all the QL software programs and in any computer application, may be the single most useful QL software utility yet produced; its uses are--to use an apt cliche--limited only by the imagination.

USING QL QUILL WITH THE NEW WONDER WEAPONS

But what do all of these high tech bells and whistles mean to the QL Quill user? Let's face it, Margaret Mitchell could have written Gone With the Wind on the T/S 2068 using Taseword II™. Are they just interesting gadgets which we would use infrequently? That is all up to you. What this new hardware and software does is permit you to realize the original promise of the QL, that is, the multitasking of two or more programs.

Uncle Clive once said that the stockQL had all of the memory and mass data storage that most PC users would ever need. But many early QL users soon found that the QL software programs--especially QL Quill and QL Archive--did not operate well on the available RAM, and increased RAM was found to truly enhance the operation of these programs. Then came the early disk drives. Many--though I was not one--experienced great difficulty with the QL Microdrives. (I still believe that for most applications the expanded QL with RAMdisk and only the QL Microdrives for mass data storage is quite satisfactory.)

The new, full-featured, disk drives with increased RAM, additional ROM utilities, parallel interfaces, and--in the case of the SuperQBoard--even more advanced ROM utilities and a "mouse" interface greatly increase the capabilities and flexibility of QL Quill and other programs. A single disk drive may have the storagage capacity of more than six Microdrives: all four QL software programs and many dozens of datafiles of all types could be put on-line for rapid loading. A second disk drive, especially with an expanded QL with RAMdisk software, gives you almost all of the quick-reaction and data storage capabilities once found only on a minicomputer.

Automatic and efficient QfLash RAMdisk software (there is no need to format or specify the size of a RAMdisk which enlarges or shrinks in capacity as data is added or deleted) provides eight more data/program banks, to and from which data can be transferred almost instantaneously. Its accompanying QfLash Toolkit which, among a great many other things, lets you load a full Microdrive into a RAMdisk in 7-14 seconds and selectively save RAMdisk files to Microdrive (that is, automatically "save" only those programs which have been altered since the last "backup" operation). The same MDVLOAD command that loads a Microdrive so rapidly into RAMdisk may load corrupted Microdrive data that cannot be loaded in any other way. This in conjunction with another QfLash utility, RAMDOCTOR_bas, might permit you to recover corrupted data salvageable in no other way. Another QfLash Toolkit command permits the rapid and automatic comparison of program or data files, bit by bit, and the correction or alteration of any undesired or corrupting segments.

There are numerous switching programs on the market now, but Taskmaster is--right now, at least--the best of the lot. Taskmaster lets you run all four QL software programs as if they were on different machines! While typing a report in QL Quill, you may switch to QL Archive or Abacus to check a needed bit of data, then, while printing a long document--perhaps from a spooler, switch to QL Easel or SuperBASIC to perform other operations. This is the way one works with a more fully integrated suite of software programs or on a minicomputer.

The 41 user definable keys in quill_key, the QL Quill version of KEYDEFINE, each representing up to 2048 characters of text or commands, may be used for many purposes: to order a simple or complex command sequence with fewer keystrokes; to insert blocks of text into QL Quill documents with only a double keystroke; and to insert research data such as a direct quotation (taken down using quill_key itself) into a formal document. If a great deal of data must be on tap, several quill_key programs may be used to hold all of it, but only one quill_key program may run at any one time.

I believe that you would agree that all of the new "wonder weapons" described have many legitimate uses with QL Quill and other programs. But there is one more essential ingredient necessary to use them all

together: a comprehensive "boot" program. I am still working on this aspect of the problem but promise to include one in a future article and in my book.

MORE QL QUILL/WORD PROCESSOR TIPS

As we complete this series of articles on word processing with QL Quill, I wonder what else out of an immense pile of notes on the subject should be included. Here goes.

If you use QL Microdrives for data storage, you will find it quicker and more trouble free to perform your file operations outside of QL Quill in SuperBASIC mode, especially if you employ RAMdisks and have the Toolkit II "wildcard" (WCOPY, WDEL, WDIR, and RENAME) or QfLash Toolkit (MDVLOAD, SBACKUP, and FBACKUP) commands available. (If you're fortunate enough to have TASKMASTER, its FILES utility is excellent for this.)

When executed, QL QUILL attempts to reserve nearly all of the available RAM for itself--it needs 60K or more data space for effective operation. If you want to run other programs with QL Quill loaded in RAM, you must reserve the space. Insert one or more rogram lines like the following in your QL Quill "boot" program for each 32K of RAM you wish to reserve outside of the program

```
dummy$=FILL$(" ",32000)
```

Insert CLEAR on the program line following EXEC_W MDV1_Quill in the "boot" program.

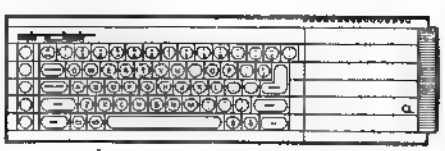
Finally, did you know that QL Quill command sequences operate much faster with the control area removed (by keying F2). The Hyphenate command takes 7-8 seconds less time to function without the control area.

NEXT TIME: QL ABACUS/SPREADSHEET TIPS and more exotic new equipment.

```
20000 PRINT "Hit S to stop, I for ICE, or any other to run JOS"
20010 ink$=INKEY$(-1)
20020 IF ink$=="s" THEN STOP
20030 IF ink$=="i" THEN ICE
20040 WHEN ERRor :IF ERN M= 19 THEN CONTINUE, END WHEN
20050 cc:CSIZE 2,1:PRINT " *** JOS ACTIVE ***"
20060 PRINT                "@1986  by JOE NEWMAN"
20070 CSIZE 0,0:PRINT "Enter year or ENTER to quit" CSIZE 3,1 INPUT "year (yy)";
 year$:IF CODE(year$)<>0 THEN year=year$ INPUT "mont ",month INPUT "day" ,day IN
PUT "hour",hr:INPUT "Min ",min:year="10"&year:SDATE year,month,day,hr,min,0
20080 CSIZE 0,0:BEEP 0,1,255,1300,1
20090 drive$="flp1_"
20100 PAUSE 50:BEEP:PRINT  "hit > i)ce or b)asic"
20110 ink$=INKEY$(-1)
20120 IF ink$=="i" THEN ICE:ELSE cc:STOP
20130 DEFine PROCedure cc:CLS#0:CLS#2 CLS:END DEFine cc
20140 DEFine PROCedure l:LIST:END DEFine l
20150 DEFine PROCedure ld (file$)
20160   call$=file$
20170   file_name$=drive$&call$
20180   MERGE file_name$
20190 END DEFine ld
20200 DEFine PROCedure default (drive$)
20210   drive$=drive$
20220 END DEFine default
20230 DEFine PROCedure LRUN (file$)
20240   call$=file$
20245   NEW
20250   file_name$=drive$&call$:MERGE file_name$
20260   RUN
20270 END DEFine LRUN
20280 DEFine PROCedure llist
20290   OPEN #100,ser
20300   cc:PRINT "READY PRINTER (hit any key)":PAUSE
20310   cc
20320   LIST #100
20330   CLOSE #100
20340 DEFine PROCedure cat
20350   DIR drive$
20360 END DEFine cat
20370 DEFine PROCedure s (file$)
20380   call$=file$:SAVE drive$&call$
20390 END DEFine s
20400 DEFine PROCedure lprint (printer$)
20410   OPEN #100,ser:PRINT #100,printer$:CLOSE #100
20420 END DEFine lprint
20430 DEFine PROCedure d (file$)
20440   DELETE drive$&file$
20450 END DEFine d
20460 DEFine PROCedure ds (file$)
20470   call$=file$
20480   DELETE drive$&file$
20490   SAVE drive$&file$
20500 END DEFine ds
20510 DEFine PROCedure screen (x$)
20520   LBYTES drive$&x$,131072
20530 END DEFine screen
20540 DEFine PROCedure commands
20550   cc:PRINT "AVAILABLE COMMANDS FOR JOS"
20560   PRINT "-------- ------  ---- -------"
20570   RESTORE 20580:FOR LOOPER=1 TO 14:READ COM$ PRINT COM$;:NEXT LOOPER
20580   DATA "LD","S","D","DS","SCREEN","CAT","LPRINT","DEFAULT","LRUN","LLIST"
"CC","L" "SIREN","NEW"
20590 END DEFine commands
20600 DEFine PROCedure siren:BEEP 0,1 1000,150,1 END DEFine siren
20610 DEFine PROCedure NEW
20620  DLINE TO 19999
20630  cc
20640 END DEFine NEW
```

Correction for JOS

By Joe Newman



After trying to run my JOS program which was presented in the Jan/Feb '87 issue of TDM (when you tried to use the LD, LRUN, or DS commands), the program most likely stopped with a mysterious error report--"not implemented". The error number is 19, which happens to be listed on page 19 of the Concepts section of the QL Users Guide, but there is no reference to what it means. Apparently the MERGE and MRUN commands cannot be called from inside a procedure.

The problem is that I left one line out of the listing. My sincerest apologies to you for any inconvenience caused by mistake. The new listing I have included not only has the "not implemented" bug fix, but a NEW command as well (which I stated in the last article would not work). It works now.

curry computer

See you at the Computer Fest in Indianapolis-Call for Show Specials.

QL SOFTWARE

GraphiQL	$49.95
Cartridge Doctor	$23.95
Zkul	$24.95
West	$24.95
Chess	$34.95
QL Art	$22.95
QDumps	$15.95
Tax-I-QL	$15.95
Qspell	$39.95
Hyperdrive	$19.95
I.C.E.	$34.95
Key Define	$17.95
QCalc	$17.95
Squadrons	$29.95
3D Slime	$19.95
R_Windows	$16.95
Bounder	$15.95
BJ in 3D Land	$19.95
Assemb. Wrkbnch	$39.95
TechniQL	$69.95
Concept 3D	$39.95
Wanderer	$34.95
Mail Merge	$24.95
Presidents	$19.95
Mortville Manor	$34.95
Peintre	$34.95
Nucleon	$27.95
Othello	$24.95
Screen.Dump	$15.95
Dragonhold	$24.95
Lost Pharaoh	$24.95
Blocklands	$15.95
Toolkit II (MDV)	$69.95
QLterm	$19.95
QCode	$29.95
Boot 128K	$11.95
Graphic Toolkit	$29.95
ICEcicle	$27.95
Swopper	$29.95
1:1 Dump	$11.95
Front Page	$37.95

QL HARDWARE

QL Computer	$199.95
Monochrome Cable	$14.95
Talker	$64.95
Cumana Disk I/F	$149.95
512K Memory	$149.95
Modapter	$59.95

QL ACCESSORIES

MDV Case/12	$2.50
MDV Case/20	$10.95
MDV Case/4	$5.95
Mini-Vac	$19.95
QL Dust Cover	$11.95
QL Print Dst Cvr	$16.95

QL BOOKS

Service Manual	$19.95
Ill. SuperBasic	$11.95
Adv Programming	$12.95
Word Processing	$12.95
Intro SuperBasic	$12.95
S.B. Handbook	$16.95
Using Graphics	$12.95
Profiting From	$12.95
M/C Programming	$12.95
Making Most of Q	$12.95
Adv. Users Guide	$29.95
Quantum Theory	$11.95
Games Master	$12.95
Advent. Handbook	$9.95
QL Computing	$9.95
QL SuperBasic	$11.95

(Add .75 s/h per book Unless ordering s/w too.)

* We are now Stocking PARTS...Give us a Call

> All S/W shipped Free. Orders over $50 shipped UPS 2nd Day Air Free (Cont. U.S. only).
> Write for a Free Catalog of QL Software & Hardware
> SHIPPING INFO: QL=$8; Mono Cable=$1; other Hardware=$2
..Accessories=$1

P.O. Box 5607 • Glendale, Arizona 85312-5607 • 1-602-978-2902 • Telex (via WUI): 6501267701

OFFICE: 2412 West Greenway • Suite B-10 • Phoenix, Arizona

curry computer

SPECTRUM SOFTWARE		SPECTRUM S/W cont		SPECTRUM S/W cont		TIMEX 1000 SOFTWARE (see)) below)	
Star Glider	$24.95	Alien Highway	$15.95	Raid Over Moscow	$15.95	Multiple Choice	$4.00
Xevious	$15.95	Sam. Fox Poker	$16.95	Jawz	$5.00	Forecast/Graph	$4.00
Lightforce	$16.95	Stairless Steel	$15.95	Dictator	$5.00	2040 Paper/3 Pak	$6.95
Gift Of The Gods	$9.95	A.C.E.	$17.95	Infiltrator	$17.95	2040 Printer	$69.95
Dlan	$14.95	Spin Dizzy	$17.95	They Sold A Mill	$18.95	TS Destroyer	$4.00
Jersey Quest	$12.95	Football Mgr	$15.95	Fourth Protocol	$19.95	Matrix Planner	$4.00
Khotoni Wilf	$9.95	Spitfire 40	$9.95	Tau Ceti	$17.95	Missle Launcher	$4.00
Knightlore	$15.95	Dambusters	$18.95	Fi te Path 737	$5.00	Textwriter 1000	$11.95
Zombie	$14.95	Delta Wing	$9.95	Frog Run	$5.00	Invent. Control	$4.00
Strontium Dog	$5.00	1 Man & Droid	$9.95	Rambo	$14.95	Algebra I	$4.00
3-D Strike Attac	$12.95	Chiller	$9.95	Wham	$18.95	Organizer	$4.00
Rebel Star Raide	$5.95	Skyranger	$12.95	Gauntlet	$15.95	Fort. Zorlac	$4.00
The Rats	$14.95	Action Reflex	$14.95	White Lightning	$22.95	Munchees	$4.00
Green Berets	$14.95	Psi Chess	$18.95	Elite	$28.95	Croaka-Crawla	$4.00
Cauldron	$14.95	Xarq	$18.95	Red Arrows	$14.95	Supernaze	$4.00
Make A Chip	$5.95	Bomb Jack	$14.95	Hypaball	$17.95	Pioneer Trail	$4.00
Glass	$9.95	Rescue/Fractalus	$16.95	Frankie/Hollywoo	$19.95	Mix Gn Bag 1	$4.00
I, of the Mask	$16.95	Hijack	$17.95	Tomahawk	$16.95	Super Math	$4.00
Robin/Sherwd	$14.95	SpeedKing J/S	$17.95	Pub Games	$17.95	Hm Asset Mgr	$4.00
Zaxxon	$12.95	Mandragore	$24.95	Sacred Armour/An	$15.95	Hangman	$4.00
Starion	$5.00	D.T. SuperTest	$14.95	Xeno	$15.95	2K Games Pak	$4.00
Night Gunner	$12.95	Toadrunner	$16.95	Gal. Abductors	$5.00	Shark Treasure	$4.00
Beach Head	$15.95	Kung Fu Master	$14.95	Commando	$14.95	Adventure Pak	$4.00
Way of Exp Fist	$15.95	Cauldron 2	$16.95	Riddlers Den	$12.95	The Fast One	$4.00
Astro Clone	$14.95	Jack the Nipper	$14.95	Skyfox	$16.95	Inventory Contro	$4.00
Fred	$9.95	1942	$15.95	Twister	$14.95	Labyrinth	$4.00
Road Toad	$12.95	Paperboy	$16.95	OmniCalc	$24.95	Pilot	$4.00
Hobbit	$19.95	Heavy 7 Magick	$16.95			Graphic Golf	$4.00
Snowman	$9.95	Boggit	$15.95	TIMEX 1000 SOFTWARE		Crossword	$4.00
Ant Attack	$5.00	Biggles	$18.95	(See)) below)		Anagrams	$4.00
Confuzion	$9.95	Wrld Ser. Bsktbl	$14.95			Red Alert	$4.00
Chequered Flag	$9.95	Ghosts & Goblins	$14.95	Meteorites	$4.00	The Gambler	$4.00
3D Tank Dual	$5.00	Bobby Bearing	$14.95	Night Gunner	$4.00	Pinball	$4.00
The Artist	$24.95	Dandy	$14.95	TI99 Keyboard	$18.95	Backgammon	$4.00
Mined Out	$5.00	Theatre Europe	$17.95	Alien Invasion	$4.00	Kasino Kraps	$4.00
Sherlock Holmes	$19.95	Aftershock	$16.95	Fin Mgr & Rec Ke	$4.00	Strategy Footbal	$4.00
Spectext	$9.95	Dan Dare	$17.95	Stock Mkt Calc	$4.00	Frogger	$4.00
Centi-Bug	$9.95	Deactivators	$15.95	Space Commando	$4.00	Heating Sys.Anal	$4.00
Traxx	$9.95	Trivial Pursuit	$22.95	Stock Market Cal	$4.00	Checkbook	$4.00
Sabrewulf	$14.95	T.P. Young Plyrs	$22.95	Chess	$4.00	Real Estate Anal	$4.00
Dynamite Dan	$14.95	Hardball	$15.95	The Puzzler	$4.00	College Cost Ana	$4.00
Terrormolinos	$14.95	Firelord	$15.95	Ram Runner	$4.00	Coupon Manager	$4.00
Lords of Midnigh	$16.95	Galvan	$14.95	Big Flap Attack	$4.00	Keyboard Calc	$4.00
Great Space Race	$5.00	Prodigy	$14.95	Tarot	$4.00	Much More-Write for List	

* Quantities may be limited on some items...State 2nd Choices.
)) 1000 Software ONLY: 2 for $6; 3 for $8; 4 for $10...all Sold 'AS IS'.

P.O. Box 5607 • Glendale, Arizona 85312-5607 • 1-602-978-2902 • Telex (via WUI): 6501267701

DESKTOP PUBLISHING!! NOW FOR THE 2068!! WITH PIXEL PRINT

PROGRAM SPECIFICATIONS:

EXTRA RAM PROVIDED FOR QL with FRONT PAGE!

- FULL SCREEN EDITING
- PRECISION LAYOUTS
- UNDERLINE FUNCTION
- VARIABLE SIZE CHARACTERS- HEIGHT TO 22 TIMES NORMAL, WIDTH TO 32 TIMES NORMAL
- VARIETY OF FONTS- STANDARD, BOLD, RUBBER, ITALICS, CURRENCY, OR LOAD A FONT
- LOAD/PRINT GRAPHICS FROM OTHER PROGS
- SUPPORTS- TS 2040 PRINTER OR ANY IBM/EPSON COMPATIBLE FOR FULL SIZE AND THE TOSHIBA (R & C) CPI, NLJ CPI, BERCO CPI AND OLICER CP!

Pixel Print was created by Stan Leake of Pixel Sketch & Graphics Editor Fame...so you know it is good! Pixel Print creates Text and loads graphics for newsletters, magazine columns, letters, etc. Pixel Print also has many powerful functions...like pixel shift for precision layouts! Pixel Print has a wide selection of fonts & sizes! Prints up to two columns wide on an 8.5" x 11" paper or a single column on the TS-2040 or a double width column on a full-size printer!

ONLY $19.95 for desk-top publishing on the TS2068!

192 x 32 COLOR REZ!! ADD COLOR TO OLD B&W EXTENDED PAINT 2068!

- Create PURELY COLOR image with a resolution of 192x32!
- OR create purely color image with a res. of 192x64 (NO Flash or Bright)
- OR ADD HIGH RESOLUTION COLOR TO EXISTING 192X256 RES. BLACK & WHITE IMAGES!
- The ONLY program to allow EASY use and access to the FULL EXTENDED COLOR MODE! Add color to Tech-Draw or Vu-3D images etc.

Comes complete with 24 page manual with examples and how to use EXTENDED COLOR MODE in YOUR programs! EXTENDED PAINT has over 25 unique commands too!

Only $19.95!

SINCLAIR SURVIVES!

2068 SPECIALS:

>TORNADO SUPER HELICOPTER SIMULATION ONLY $15.95 POST PAID TILL APRIL 1!!

>MACHINE CODE TUTOR-33 LESSONS, 2 TAPES ONLY $17.95 POST PAID TILL APRIL 1!!

>>>> NEW METAL CASED MICRODRIVES!! <<<<

as CORRECT, ATTRACTIVE METAL CASE as READ/WRITE LED, THIS DRIVE LOOKS LIKE A MINI HARD DISK DRIVE...BUILT TO LAST

A&J MODEL 2000 ONLY $139 !

PLEASE ADD $6.50 SHIPPING FOR MICRODRIVES. INCLUDES 5 FREE MICROCARTS, MANUAL, INTERFACE, DRIVE & CABLE

TEL: 201-527-0535

FRONT PAGE

BY GAP

DESK-TOP PUBLISHING FOR THE SINCLAIR QL

ONLY- $34.95 PP!!

Version 1.01 now in! Here is a list of the improvements:

Supplied on 1 cartridge! You can now load in your own character sets! Even from Nucleus or QL World! You can now add text OVER existing text/graphics, without wiping them out! Toggle between character sets while inputting text! Also supports PARALLEL printer interfaces!

Above stolen from QL WORLD mag., Aug. '85, from a program by ALAN PEIER... Screen was then loaded into FRONT PAGE and printed with the QL PRINTER. Nice!

64 COLUMN TERMINAL PROGRAM! NOW FOR THE 2068 & SPECTRUM!! NO EXTRA SOFTWARE or HARDWARE NEEDED!

You can either run the 2050 modem or use an RS232 Board and a 1200 baud modem with this software! Supports XMODEM too!!

SPEC TERM-64...the ONLY 64 column terminal software for the 2068 which requires NO extra hardware or software! Has X-modem up/downloading, supports any 2068 RS232 board with RS232 modem, can up/download ANY bytes or program file EASILY with no extra steps! Automatically saves text files in the format for Tasword! Has a 7K area for your OWN CUSTOM" menus, over 31K buffer area for files! Also allows text transmittal as ASCII (stops at codes over 127). SPECTERM-64 can support 4 cmdrives, disk, etc. FREE support and EXTRA FEATURE menus are available 24 hrs. on the TIME-CODE-CHANGE BBS.

TWO VERSIONS, PLEASE SPECIFY: Spectrum version (2068 with emulator), or 2068 only. The two are NOT interchangeable! Please order carefully!

Only $34.95 Post Paid!

VARIETY SALES DEPARTMENT TDN-4

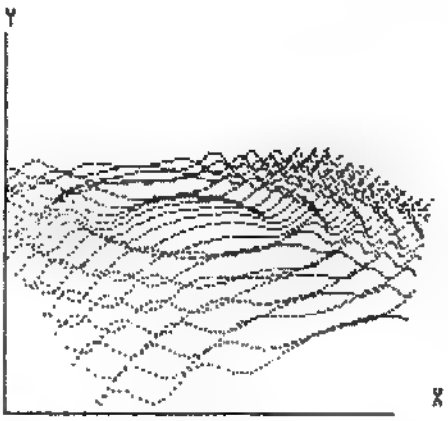
325 W. JERSEY ST.,#2D ELIZABETH, NJ 07202

ADD $3.00 TO ORDER FOR FOREIGN AIR MAIL POSTAGE AND $5 SHIPPING FOR SOFTWARE UNLESS SPECIFIED OTHERWISE!!

```
100 REMark CRATER for QL, Gale Hensles, 1987
110 MODE 4:PAPER #2,0 BORDER #2,0,0:CLS #2:PAPER #1,0:BORDER #1,0,0:CLS #1
:WINDOW #1 410,160,46,0:CLS #1
120 INK #1,4:LINE 0,0 TO 100,0:LINE 0,0 TO 0,100:CURSOR 200,148:PRINT #1,"
X":CURSOR 0,0:PRINT #1,"Y":INK #1,4
130 q=-4
140 FOR a=23 TO -16 STEP -1.6
150     IF a>5 :q=q-1.7
160     ELSE q=q+1:END IF
170 FOR b=q TO 15 STEP .3
180     z=COS(6E-2*(a*a+b*b))
190     h=80-a*.9-b-z
200     x=49-250*((-a*.44+b*.9)/h)
210     y=49+250*((-a*.3-b*.14+z*.7)/h)
220         IF x>200 OR x<0 THEN NEXT b
230     POINT #1, x,y
240  END FOR b
250  END FOR a
260 CLOSE #2 CLOSE #1
270 CLS #0:PRINT #0,"Press any key when ready to exit."
280 PAUSE
290 REMark test for monitor or TV mode
300 TV_VALUE=PEEK(163880)
310 IF TV_VALUE=2:GO TO 380
320 REMark reset monitor mode screens
330 OPEN #2,scr:WINDOW #2,256,200,0,0:PAPER #2,3:INK #2,6:BORDER #2,2,0,7:
CLS #2
340 OPEN #1,scr:WINDOW #1,256,200,254,0:PAPER #1,6:INK #1,3:BORDER #1,2,0,
7:CLS #1
350 WINDOW #0,512,57,0,188:PAPER #0,0:INK#0,5:CLS #0
360 STOP
370 REMark reset TV mode screens
380 OPEN #2,scr:WINDOW #2,420,160,46,0:PAPER #2,3:INK #2,6:BORDER #2,2,0,7
:CLS #2
390 OPEN #1,scr:WINDOW #1,420,160,46,0:PAPER #1,6:INK #1,3:BORDER #1,2,0,7
:CLS #1
400 WINDOW #0,420,32,46,160:PAPER #0,0:INK#0,5:CLS #0
410 STOP
```

QL CRATER

By Gale Hensles



WMJ DATA SYSTEMS

Serving the Timex/Sinclair community since 1983

- Free shipping on all prepaid orders!
- Shipment within 24 hours on most of our orders!
- Your personal check is gladly accepted with no delay!
- C.O.D. phone orders accepted! (Shipping charges added)

TS1000

ACZ GENERAL LEDGER	TS1000	$19.95
SUPERTAPE-45 PROGRAMS	TS1000	18.95
TEXTWRITER 1000-32 col.	TS1000	12.95
SPEECH SYNTHESIZER	TS1000	16.95
STOCK PLOT	TS1000	14.00

BOOKS AND ACCESSORIES

KRAFT SYSTEM JOYSTICKS	TS2068	$ 9.95
HACKER'S HANDBOOK	ALL	9.95
BASICS AND BEYOND	TS2068	7.00
C10 CASSETTES	25 for	21.00
C20 CASSETTES	25 for	23.00

TS2068

ACZ GENERAL LEDGER	TS2068	$19.95
THE DEALER'S DEN	TS2068	19.95
MACHINE CODE TUTOR	TS2068	18.50
TOMAHAWK	TS2068	16.95
NIGHT GUNNER	TS2068	16.95
FIGHTER PILOT	TS2068	16.95
CRITICAL MASS	TS2068	15.95
SABOTEUR	TS2068	15.95
BASIC TOOLKIT	TS2068	13.95
MSCRIPT (specify conf.)	TS2068	24.95
DIAMOND MIKE	TS2068	16.95
COLONIZE THE UNIVERSE	TS2068	16.95
STOCK PLOT	TS2068	16.00
All QUICKSILVA Software	TS2068	CALL

WMJ DATA SYSTEMS—TD

4 Butterfly Drive
Hauppauge, NY 11788

(516) 543-5252 (2 p.m. - 5 p.m. Best)

FREE Apple, Atari, Commodore, IBM...Catalog
FREE Timex/Sinclair Catalog - Write or Give us a CALL!

RESOURCES FOR YOUR SINCLAIR
FROM TIME DESIGNS MAGAZINE COMPANY

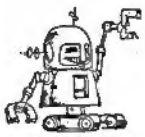
GREAT BOOKS FOR YOUR SINCLAIR

OFFICIAL 2068 TECHNICAL MANUAL.....$25.00

CONTROL THINGS with your Timex/Sinclair by R.L. Swarts.....$5.00

ZX81: Programming for Real Applications by Randle Hurley.....$6.00

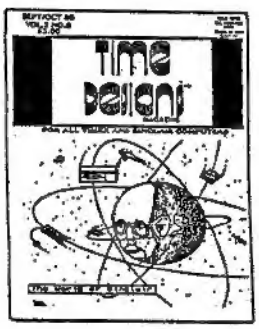
Includes 12 Software Programs on a Cassette Tape



Build a Microcomputer - Controlled Robot by B.C. Taylor.....$17.00

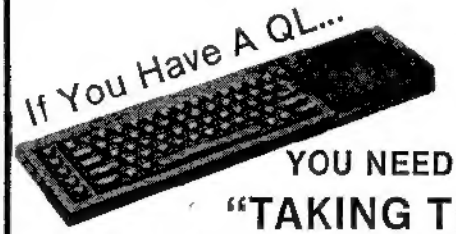
BOTH *"Control Things"* and "ZX81" Package For Only **$10.00**

TIME DESIGNS MAGAZINE



Still Only $15 Year Six Issues

TRY A SUBSCRIPTION TODAY!



SEE THIS NEW BOOK AT THE FEST!

YOU NEED

"TAKING THE QUANTUM LEAP"

By Mike de Sosa

The Ultimate Resource For Your Sinclair QL

AVAILABLE SOON!

Published Exclusively By TIME DESIGNS
Write or Call for Information

TIME DESIGNS MAGAZINE BACK ISSUES $3.00 EACH

Nov/Dec 85 VOL.2

Jan/Feb 86

March/April 86

May/June 86

July/August 86

Sept/Oct 86

Nov/Dec 86 VOL. 3

Volume One Issues are no longer available individually (See "BEST OF TDM").

The Best of TIME DESIGNS Volume 1

$10.00

QTY	ITEM		AMOUNT
	TIME DESIGNS MAGAZINE Subscription - One Year	$15	
	TIME DESIGNS BACK ISSUES Specify Which:	$3.00 each	
	THE BEST OF TDM Vol. 1	$10	
	TS 2068 Technical Manual	$25	
	Control Things With Your Timex Sinclair	$5	
	ZX81: Programming for Real Applications (includes tape)	$6	
	SPECIAL PACKAGE DEAL: Both "Control Things" and ZX81"	$10	
	Build a Microcomputer Controlled Robot	$17	
	Taking The Quantum Leap	Write or Call	
•	WE PAY THE POSTAGE ON ALL ORDERS!	TOTAL	

FOR MAGAZINE SUBSCRIPTION (please circle one below):
NEW SUBSCRIPTION THIS IS A RENEWAL THIS IS A GIFT

SHIP TO:

NAME_____

ADDRESS_____

CITY/STATE_____

ZIP_____

PHONE_____

PLEASE PRINT NEATLY.

CREDIT CARD
(Circle One)
MASTERCARD / VISA

Acct. Number_____

Exp. Date_____

Signature_____

Note: 4% charge added to all Credit Card Orders.

PAYMENT
Check or M/O.

VISA and MASTERCARD ACCEPTED

VISA MasterCard

24 hour ORDER LINE
(503) 824-2658
For Books, Subscriptions and Renewals

time designs

TIME DESIGNS MAGAZINE CO.

29722 Hult Rd., Colton, OR 97017

the CLASSIFIED

FREE ADS FOR SUBSCRIBERS

One TASMAN TS2068 CENTRONICS I.F. with cable & driver SW for parallel printers (original cost $89.95) for $55.00 ppd. One like-new copy of 500-page GUIDE TO T/S COMPUTERS by Morse, Adamson, Anrep, Hancock for $9.00 ppd.--an excellent BASIC text-book. R.D. Hartung, 2416 N. County Line Rd., Huntertown, IN 46748.

FOR SALE: TS2068 (new), still in the original box. Also ZX81 and TS1000 with Rampacks, power supplies and manuals. Dick Taylor, 309 Holly Circle, Tiverton, RI 02878. (401) 624-2035.

FOR SALE: QL computer, printer, various books, spare power supplies, and more. Call for prices and details. (603) 352-9836.

SINCUS NEWS-$8/yr=6 issues. TS2068 oriented user group newsletter. Mail check to SINCUS, 1229 Rhodes Rd., Johnson City, NY 13790: a non-profit amateur users-oriented group.

WANTED: A legal copy of MICRO-PROLOG for the Spectrum, complete with original docs. Willing to pay retail price. Also wish to contact others working with it. Pete Fischer, POB 2002, Tempe, AZ 85281.

FOR SALE: TS2068, power supply, 2050 modem, interface, assorted software/cartridges, $175; Olevetti PR2300 dot matrix printer, orig. $250, only $125. Contact: Paul Maxwell, 8023 Park Lane, Bethesda, MD 20814 or phone (202) 225-8115.

WANTED: ZX99, HOT-Z, BYTE-BACK, or other add-ons and utilities for the TS1000. Also want Syncware News back issues. Wanna swap software? Send me your list. Dan Elliott, Rt 1 Box 117, Cabool, MO 65689.

WANTED: SINCLAIR DIGITAL FM RADIO WRISTWATCH in good condition. Also want working and good condition 48K ZX Spectrum (with rubber keys). Send reply to: Tim Woods, c/o TDM, 29722 Hult Rd., Colton, OR 97017.

TIMEX 2068 "DESKTOP PUBLISHER" Software. $19.95 + 1.25 for post. and handling. Charles Stelding, 1415 S. Baxter, Tyler, TX 75701.

WANTED: TS2068 Owner's Manual. John Sandgren, PO Box 539, Cavalier, ND 58220.

MONTHLY PLANNER Pgm for TS1000 and 1500. Ergonomic software keeps track of Events/Dates--Fun to use, micro drive compat! Only $11. M. Felerski, 2110 Parnell Ave., Ft. Wayne, IN 46805.

SOFTWARE FOR THE TS2068. Send SASE for free catalog & price list to: E. Ray Rash, 2424 S.W. 78th St., Okla. City, OK 73159.

WANTED: BOOKS AND MAGAZINES for TS2068. Send list with prices to: Arthur V. Murray, 99 Waverly Ave., Apt. 9-C, Patchogue, NY 11772.

WANTED: TS2068 COMPUTER, new or used, but complete and in good condition. Call or write: Tony Willing, P.O. Box 199, Vashon, WA 98070. Phone-(206) 567-4118 eves. or (206) 463-3682 weekdays.

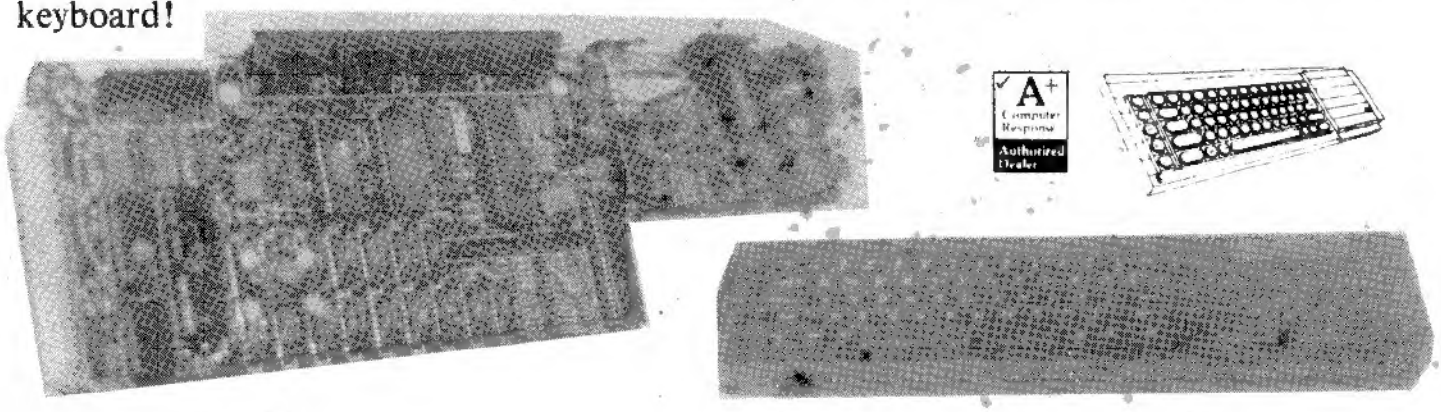
FOR SALE: OVER 30 SPECTRUM and TS2068 programs for as low as $8 each (U.S.) incl. p&h. Some titles: Ghostbusters, Commando. Send SASE to: Alvin Albrecht, Box 2513, Medley, AB, Canada T0A 2M0.

Do you have some equipment or a program that you would like to sell? Looking for something hard to find? Place an ad in **THE CLASSIFIEDS!** Subscribers can place one free personal ad in each issue. Ad size is 32 Col. wide (like 2040 paper) and maximum of six lines. For additional lines - $3 each. **NON-SUBSCRIBERS** and **DEALERS:** $4 a line. **DEADLINE FOR ALL CLASSIFIED ADS:** Two weeks before publication date. Mail your ad to: **TIME DESIGNS MAGAZINE, The Classifieds Dept., 29722 Hult Rd., Colton, Oregon 97017.**

Sinclair Users: You know without a doubt what the power of Sinclair Superiority is or you wouldn't have these pages in front of you!!!

You know how Sinclair Computing just leaves all of those other "Status Quo Sheep" PC's, Cloneys, etc. in the DUST!

The QL takes this Sinclair Superiority Tradition right up to it's highest form using 68000 architecture! The QDOS & SuperBasic Environment constitute the Greatest Computer Achievement of the 20th Century!! Like all of the prior Timex implementations, you have to use the computer to appreciate the beauty all locked up inside of the keyboard!



Dedicated Usership:
you are one of the users who is still hugging the prior superior computing models of ZX-81 and 2068, don't you think that now is the TIME to get yourself one of these ORIGINAL Sinclair QL Designs? It won't be long now before other updated, enhanced U.S. QL models go into production. This will most likely cost from 50% to 100% more than the original QL because of the expenses involved in implementing such things into production. We have been selling the QL for **well below retail.** Don't wait until future models reduce the current original version to a mere "Collector's Item".

Longevity For Once:
The Sinclair QL Environment will NEVER be abandoned like the Timex versions. There is too much power contained in the design, and enough other companies are involved in compatibles and high level software!

Special To All Time Designs Readers:
Buy Your QL Now For $175.00!! plus $12 ship. & handl.

A Flat Price for TD Readers with TS 1000, 1500, or 2068. Finish your collection! Have all of the original Sinclair Research Laboratories family of miniaturized computer marvels!

Call: (201) 328-8846
QLine BBS: 328-2919

Technical Info & Assistance - -
Our Phones are Manned 24 Hours a Day
Lot Purchases Available. We Direct Distribute.

Quantum Computing, Box 1280, Dover, NJ 07801

WEYMIL presents...

A small collection of truly innovative products for Sinclair computers

NEW FOR 1987 !!

MULTI-TASKING ON THE TS 1000?

BELIEVE IT !!!

WEYMIL continues to deliver what everyone else says is "impossible." What's the hottest feature in computers today? MULTI-TASKING! Until now, most people thought that you had to buy an expensive 68000 machine for multi-tasking. Please allow us to introduce NOVA 1000, the software that delivers MULTI-TASKING to the TS 1000. No gimmicks, no tricks. A full multi-tasking environment for the TS 1000 at a very affordable price. Our developers have outdone themselves on this one. Please read on.

NOVA 1000 comes complete with these powerful built-in features. An honest-to-goodness switchable real-time clock that runs concurrent with your programs ON THE SCREEN. A switchable automatic line tracer to allow you to follow the operation of your BASIC programs. Switchable auto-repeat on all keys, just the thing for creating long REM statements. There's more.

Each copy of NOVA 1000 comes complete with 3 complete versions. Version 10 contains the complete NOVA 1000 program in statement 0 REM to allow you to incorporate its features in many existing programs or programs of your own design. Version 20 automatically locates itself to the 8 to 16K region for Hunter Board or 64K applications. Version 30 automatically lowers RAMTOP for 16K configurations.

Consider these possible applications of NOVA 1000. A background auto-dialer for BBS applications would allow your computer to be used for other purposes while waiting for access. The real-time display could be used to keep track of BBS time usage, great for pay boards or long distance calls. Multiple M/C programs can now be run simultaneously. The real-time clock can be used as a timer to activate or de-activate several programs at once. BASIC and M/C programs can be run together. Package M/C programs as memory resident BATCH files. Create printer buffers for W/P applications. Transfer data between two or more on-line programs. Control multiple real-time I/O events for powerful low-cost robotics. Monitor the actual running of BASIC programs rather than just watching the results. There's still more.

NOVA 1000 is very user-friendly and well-documented in laymen's language. You would expect a program with these features to be memory intensive, NOVA 1000 uses only 512 bytes! NOVA 1000 does not slow down your computer. SLOW mode programs can actually have speed increases up to 400%! Registered users will receive FREE periodic application update sheets.

We feel confident that NOVA 1000 will redefine the standards for TS 1000 programming. Consider this offer. If you can develop imaginative programming applications utilizing NOVA 1000, we will provide our proven marketing expertise. We will handle advertising, distribution, and customer service PLUS pay you the most competitive rates in the industry. All you have to do is create and we'll take it from there. This offer is restricted to registered users.

NOVA 1000 for the TS 1000 only $ 20.00

TRACER TS2068

The perfect de-bugging tool for BASIC programs. This program allows you to actually see each line of your BASIC program as it is executed. This program prints not only the line number in use, but the commands as well. It uses only the bottom of the screen for display rather than interrupting the screen display. De-bugging has never been easier.

TRACER FOR THE TS 2068 only $ 15.00

SOUNDESIGN TS2068

The best sound development program we've seen for the TS2068. This program allows you to manipulate the sound chip of the TS 2068 in ways never before imagined. The sound effects, synthesis, and music this program can create have to be heard to be believed. Along with built-in sound routines, this software allows you to develop and mix your own as well. Once you have created the sound you want, all you have to do is press a key and you get a print-out of the line of BASIC required for you to implement that sound in your favorite program. This program is not only a powerful utility for any software library, it is also just plain fun.

SOUNDESIGN for the TS 2068 only $ 15.00

SPECIAL OFFER!!!!

Save $ 5.00 when you order the combination of THRUST, RIGTER, JOYSTICK INTERFACE, and KRUNCHER 1000 and pay only $64.95

SHIPPING INSTRUCTIONS: Please enclose $2.00 shipping and handling with your order.

WEYMIL CORPORATION

Box 5904

Bellingham, WA 98227-5904